Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Rakshith Shetty

Natural Language Description of Images and Videos

Master's Thesis
Espoo, September 8, 2016

Supervisor: Professor Juha Karhunen
Advisor: D.Sc. (Tech.) Jorma Laaksonen

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Rakshith Shetty		
Title:	Natural Language Description of Images and Videos		
Date:	September 8, 2016	**Pages:**	95
Supervisor:	Professor Juha Karhunen		
Advisor:	D.Sc. (Tech.) Jorma Laaksonen		

Understanding visual media, i.e. images and videos, has been a cornerstone topic in computer vision research for a long time. Recently, a new task within the purview of this research area, that of automatically captioning images and videos, has garnered wide-spread interest. The task involves generating a short natural language description of an image or a video.

This thesis studies the automatic visual captioning problem in its entirety. A baseline visual captioning pipeline is examined, including its two constituent blocks, namely visual feature extraction and language modeling. We then discuss the challenges involved and the methods available to evaluate a visual captioning system. Building on this baseline model, several enhancements are proposed to improve the performance of both the visual feature extraction and the language modeling. Deep convolutional neural network based image features used in the baseline model are augmented with explicit object and scene detection features. In the case of videos, a combination of action recognition and static frame-level features are used. The long-short term memory network based language model used in the baseline is extended by introduction of an additional input channel and residual connections. Finally, an efficient ensembling technique based on a caption evaluator network is presented.

Results from extensive experiments conducted to evaluate each of the above mentioned enhancements are reported. The image and video captioning architectures proposed in this thesis achieve state-of-the-art performance on the corresponding tasks. To support these claims, results from two video captioning challenges organized over the last year are reported, both of which were won by the models presented in the thesis. We also quantitatively analyze the automatic captions generated and identify several shortcomings of the current system. After having identified the deficiencies, we briefly look at a few interesting problems which could take the automatic visual captioning research forward.

Keywords:	Image Captioning, Video Description, Deep Learning, Long-short term memory, Language Modeling
Language:	English

Acknowledgements

This thesis was completed under the supervision of Prof. Juha Karhunen, and I would like to thank him for his support.

I cannot thank enough my advisor Dr. Jorma Laaksonen for his constant support, help with the research work, the funding to carry out this research, diligent feedback and his patience with my erratic writing.

I wish to thank Dr. Hamed R.-Tavakoli for many insightful inputs which has contributed a lot to this thesis.

I would like to thank my collogues Vikram Kamath and Pablo Alonso for some very interesting discussions we had.

I acknowledge the computational resources provided by the Aalto Science-IT project. The calculations presented in this thesis were partly performed using computer resources within the Aalto University School of Science "Science-IT" project

I thank my parents and my brother for being my source of inspiration and comfort.

Finally, to Klaara, thank you for the support and motivation you provided to help me make it past the finish line.

Espoo, September 8, 2016

Rakshith Shetty

Acronyms and Symbols

Acronyms	**Expansions**
BLEU	Bilingual Evaluation Understudy
CIDEr	Consensus-based Image Description Evaluation
CNN	Convolutional Neural Network
DT	Dense Trajectories
GPU	Graphics Processing Unit
IDT	Improved Dense Trajectories
LCS	Longest Common Subsequence
LSMDC	Large-Scale Movie Description Challenge
LSTM	Long-Short Term Memory
MS-COCO	Microsoft Common Objects in Context
MSR-VTT	Microsoft Video To Text
METEOR	Metric for Evaluation of Translation with Explicit ORdering
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SVM	Support Vector Machine

Symbols	**Meaning**
b	size of the beam used in beam search
$C(\cdot)$	function mapping a word to its class

D	decoder matrix mapping LSTM hidden state to the output vocabulary
D_{cls}	decoder matrix mapping LSTM hidden state to a vector of class scores
D_c	decoder matrix corresponding to class c
F_c	grid cell corresponding to object of class c in spatial map features
I, P	*init* and *persist* inputs in the proposed language model
i, o, f	input, output and forget gates of an LSTM cell
m	memory unit in an LSTM cell
L	length of a caption in words
$\mathcal{NL}$	negative log likelihood
R, P	Recall and Precision
S	a caption or a sentence
V	visual input, an image or a video
w_i	i^{th} word in a sentence $(w_0, \cdots, w_{L-1})$
W^t	partial sequence of words upto t $(w_0, \cdots, w_t)$
$W_{ix}, W_{ox}, W_{fx}, W_{mx}$	input to gate matrices in an LSTM cell
$W_{iy}, W_{oy}, W_{fy}, W_{my}$	recurrent gate matrices in an LSTM cell
Z	number of words in the vocabulary
Z^c	number of classes the vocabulary is split into
Z_w^c	number of words in the class c

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A picture is worth a thousand words. But how many can the machine say?

Our adaption of an old English proverb

The famous English adage states "A picture is worth a thousand words". Translated into technical terms, this is meant to convey that there is a lot the viewer can learn or infer from a single still image and that enumerating all the information encoded in an image can take up to even a thousand words. This is illustrated in our extensive use of images in all forms of communication, from scientific journals to Twitter chats. Humans are very good at processing images and videos and gathering all this encoded information, but the computers still struggle to make sense of the simplest ones. One could say it is still easier for computers to store, parse, search and even understand a thousand words than a single image.

The use of multimedia on the internet has grown to staggering levels in the recent years, due to easy access to cameras in smart phones. For example about 95 million photos are uploaded to Instagram every day [2] and about 400 hours of video is uploaded to YouTube every minute [1]. Rapidly growing amount of visual data being created due to this phenomenon presents both an enormous challenge and an opportunity to build smarter computer algorithms to understand and summarize the data. Such algorithms could help us index and search this visual data better. An algorithm which can learn to recognize and describe different objects and their relationships in an image or a video would be an essential building block of a general artificial intelligence (AI) system. Hence, automatic understanding of visual media is an interesting and important problem in many aspects of computer vision and AI.

An essential research topic at the heart of machine understanding of visual media is automatic captioning of images and videos. This involves designing

an algorithm which takes the image or the video as input and generates a natural language caption succinctly describing the content of the media. Effectively solving the above problem requires the machine to be able to identify the salient objects in the image or video, recognize their attributes, extract the relationships between these objects and also to correctly recognize the scene. This machine also needs to be able to use the extracted information to generate a natural language caption summarizing the essence of it. Since the caption generation requires both visual feature extraction and natural language generation modules, it is also a good proxy task to measure the progress in both these domains.

Until recently, the task of reliably identifying even a single object in an image across diverse and large-scale datasets was hard. This changed dramatically with the availability of large-scale annotated data such as the ImageNet dataset [12], and the application of deep learning techniques, specifically convolutional neural networks (CNN). For example, in the image classification task of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [52], which involves classifying images to one of thousand object classes, the accuracy has improved from 71.8% to 95.06%, surpassing the human performance on the same task. It has also been discovered that image classification networks which are trained on the large ImageNet dataset, also generalize very well and can be used as generic image feature extraction for different tasks [74]. This has led to successful application of such deep networks to various other tasks in computer vision including the task of image captioning.

In this thesis we will examine the task of automatic image and video captioning and discuss algorithms utilizing tools from deep learning to solve this task. Much of the discussion presented in this thesis applies to both image and video captioning problems. In such cases, in the interest of conciseness, we use the term "visual captioning" to refer to both of these problems. Next, we will define the problem more precisely and list out the basic building blocks of a visual captioning pipeline.

1.1 Problem Statement

Our task is to generate a caption given an input image or video. We are going to focus on methods to generate single sentence captions only. Thus a caption can be precisely defined to be a sentence, S, which is a sequence of words $(w_0, w_1, \cdots, w_{L-1})$ with L being the length of the sentence. We are trying to learn the distribution, $P(S|V)$, where V is the visual input (either an image or a video). This can be written as

Caption: *a giraffe walking through a patch of high dried out grass.*

Figure 1.1: A sample image–caption pair from the MS-COCO dataset.

$$P(S|V) = P(w_0, w_1, \cdots, w_{L-1}|V). \tag{1.1}$$

In this work, the probability distribution, $P(S|V)$ is modeled using a two staged approach, as has been popular in the image captioning literature. In the first stage the visual input V is mapped onto one or more feature vectors V_f. This process is deterministic and the feature vectors extracted are of fixed size for every input. In this thesis, we will explore different methods for extracting feature vectors from images and videos and analyze their performance quantitatively and qualitatively on the automatic captioning task.

For images, we will experiment with features extracted from CNNs trained on ImageNet for single image classification, explicit object detector features, and features constructed from object localization networks. In the case of videos, we will explore dense trajectory features, frame-level CNN features and 3D convolutional features.

The next stage is the language model, which takes the visual feature vector V_f as input and learns a probability distribution over sentences. Since the sentences are sequence of words, they lend themselves naturally to be modeled using sequential models such as recurrent neural networks. In this thesis we will only deal with language models based on the recurrent networks, specifically a variant called Long-Short Term Memory (LSTM) [23] networks.

We will implement and analyze a popular implementation proposed in literature [65]. Then we propose some extensions to this baseline language model by adding additional input channels, using deeper networks with residual connections, and class-based factorization of the language model. We will also discuss several techniques to create an ensemble of language models, exploring the problem of picking one best caption from a pool of captions generated by each model in the ensemble.

Evaluating an image captioning system is also non-trivial, since we have to compare the generated caption against a few different reference captions. The standard recipe followed in the literature, which we also use here, is to use multiple automatic evaluation metrics adopted from the field of machine translation research. In addition, we present human evaluation results obtained by our participation in two automatic video captioning challenges. Concretely, we participated in the Large Scale Movie Description Challenge (LSMDC) 2015 and the Microsoft Video to Text Challenge (MSR-VTT) 2016 and the results from these competition will be presented here. Our video captioning models won both the competitions as judged by human evaluators.

We analyze the results of the experiments qualitatively and discuss the strengths and shortcomings of current solutions. Drawing from this analysis, we finally discuss a few promising new directions the research on vision and language is heading.

In summary, the main contributions of this thesis are as follows:

- Discuss the basic caption generation pipeline in detail.
- Propose use of alternative image and video features to improve the performance over the baseline model.
- Propose extensions to the basic language language model to improve the accuracy and diversity of the generated captions.
- Propose an effective method to ensemble multiple caption generator models to further improve the performance.
- Experiments to prove the efficacy of our proposed extensions and qualitative analysis of the captions generated.
- Identify and discuss a few promising directions of research to improve and extend the visual captioning systems.

Some content of this thesis overlaps with the three papers published related to this work. A part of the experiments on image captioning has been

published in [57]. Our video captioning solution which won the LSMDC 2015 has been published in [55]. Finally our video captioning system which won the MSR-VTT challenge has been published in [56].

1.2 Structure of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, a discussion on the background literature related to the building blocks of caption generation models, i.e. visual feature extraction and language modeling, is presented. Here we will also review the several related works on visual captioning and discuss the datasets available to train such models. The details of our baseline caption generation model and the automatic metrics used to evaluate a captioning system are discussed in Chapter 3. Chapter 4 presents some extensions to the image and video features compared to the ones used in the baseline model. Chapter 5 discusses several extensions to the baseline language model and presents some ensembling techniques to combine multiple language models. Chapter 6 contains results from several experiments to determine the best configurations for our image and video captioning systems and provide comparisons to a few other state-of-the-art models from the literature. In Chapter 7, some shortcomings of our visual captioning systems are identified and a few interesting problems to explore to address these issues are discussed. The thesis is concluded in Chapter 8.

Chapter 2

Background: Vision & Language

In this chapter we will review some background literature exploring different facets of integrating visual data and their natural language annotations. This includes learning good representations for images and videos, different kinds of generative language models, techniques which attempt to rank similarity between visual data and language annotations, and finally some recent advances in captioning images and videos. Additionally, we will review some datasets available for training such captioning models.

2.1 Visual Features

Visual media, be it image or video, are inherently very high-dimensional data. This large dimensionality poses a challenge for machine learning systems trying to extract higher-level semantic information directly from such visual inputs, as in case of captioning. To address this, images and videos have traditionally been represented with smaller feature vectors which attempt to encode the most important information present in them, while ignoring redundancies. This feature extraction step is very important in any image understanding pipeline as it usually serves as input to the subsequent modules and hence can be a major bottleneck to the performance of the entire system. Therefore, we will now review some feature extraction techniques for images and videos and identify the best performing ones, which will be used in designing the captioning system later.

2.1.1 Image Features

Traditionally, tasks such as object recognition have relied on using hand-crafted features to represent images. Recently, however, deep Convolutional

Neural Networks (CNN), which learn to extract features necessary for the task entirely from the data, have become a popular choice for image feature extraction. This was triggered by the spectacular improvement in image classification accuracy seen on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012, with the first use of CNNs in this competition. In this challenge, involving classifying the input images to one of thousand classes, the submission by Krizhevsky *et al.* [34] using a deep CNN outperformed all the others by a large margin. This set off further exploration into CNN architectures and has driven up the performance in the ImageNet classification task to even surpass the human classification accuracies [20].

More interestingly, the deep CNNs trained on the large ImageNet dataset for the classification task have been shown to generalize very well to other datasets and tasks as well. In [74], it is shown that the weights learned by the CNNs pre-trained on the ImageNet dataset are good initializers for other tasks as well. That is, if we use the weights from CNNs pre-trained on ImageNet to initialize the networks before training it on other datasets and tasks, we can learn much better models than just using random initialization. Alternatively, using activations from some higher layer of an ImageNet pre-trained CNN as off-the-shelf image features has also been shown to produce state-of-the-art results [15, 54, 56, 57] in several datasets and tasks, including object detection, scene recognition, image and video captioning, etc. We will follow this second approach, i.e. use activations from CNNs pre-trained on ImageNet as feature input to our captioning model, without any fine-tuning of the CNNs for this task. GoogLeNet [59] and VGG [58] architectures, which won the different categories of ILSVRC 2014 competition, have been popular models for such feature extraction in the community with the ready availability of the code and pre-trained models.

2.1.2 Video Features

Unlike in the case of images, where the convolutional neural network (CNN) image features have become the *de facto* standard features for many image understanding related tasks, no single video feature extraction method has achieved the best performance across tasks and datasets. Dense trajectories [66] and Improved dense trajectories [67] have been popular video feature extraction methods for the task of action recognition. In these methods, interest points are densely sampled in an initial frame and then tracked across frames, to form trajectories. Furthermore, a set of local descriptors are extracted around the trajectory coordinates to obtain a rich representation of the trajectory. These trajectories can encode motion patterns from a variety of sources, including actions from agents in the video, camera motion, etc.

Following the success of the deep CNN models on static images, there have been attempts to train 3-D CNNs which operate directly on video segments [25, 31, 61]. However, these models need a lot of training data and are usually pre-trained on some large action recognition dataset, e.g. the Sports-1M dataset [31].

All of the above features encode action-related information very well, but fail to capture information about the identity of the objects in the video. The task of caption generation also requires us to describe the objects seen in the video and their attributes, in addition to recognizing actions. This can be addressed by extracting features from individual frames [64] or key-frames [55] using CNNs pre-trained on the ImageNet dataset.

In this work we explore both these paradigms of video feature extraction, namely hand-crafted trajectory features and deep video features, for the task of automatic visual captioning.

2.2 Natural Language Modeling

Generative language models are widely used in various tasks including speech recognition and synthesis, document analysis, dialog systems, etc. Our task of caption generation also involves learning a conditional generative language model which can generate captions given the input visual features. For this purpose, we will now discuss a few different language modeling approaches, and evaluate their suitability for our task.

The simple n-gram language models, which are based on counting co-occurrence statistics of sequence of n words, are surprisingly good baselines for a lot of language Modeling tasks. However, they are constrained to generate sentences only by using n-grams they have seen in the training set. The maximum entropy language model (ME-LM) [5] overcomes this problem by using the principle of maximum entropy while learning the model. The principle dictates that among all the probabilistic models which satisfy the constraints of the training data, one should pick the model which is the most uniform. This allows the model to share some probability for unseen n-grams as well.

Both the above models suffer from using a short context of previous words when predicting the next word, limited by the n-gram size, which can lead to longer generated sentences being incoherent. Alternatively, one could use recurrent neural networks as generative language models as proposed in [41]. Recurrent neural networks have the benefit of having access to theoretically infinite context through their hidden state. Additionally, these models do not need pre-defined language features, unlike in case ME-LM, and can learn

the necessary word representations from the data. Indeed, such recurrent network based language models are quite popular in the machine translation task [3].

2.3 Intermediate Problem: Multi-Modal Embeddings

A precursor to the problem of caption generation from input visual features, is the problem of learning to map both visual features and the corresponding natural language labels into a common semantic space. This was posed as a solution to automatically annotate images in [68]. Here, using a simple bag-of-visual-words image representation, both the image and words are projected into a common space, with the training objective of ranking correctly the matching image and annotation pairs.

In [18], using the CNN features as the image representation and the word vectors from word2vec [42] embeddings as the word representation, a linear embedding is learned to map the image vectors onto word vectors corresponding to the labels associated with the image. This allows the model to do "zero-shot" recognition of image classes it has not seen before, by finding the nearest label to the embedded image feature. A much simpler approach is used in [43], where the mapping to semantic embedding space is done by a convex combinations of c word vectors associated with the top c classes identified in the image. This does away with the need to learn the embedding, while still achieving impressive results in zero-shot classification on the ImageNet dataset.

The methods discussed above forms the basis for the way visual features are used in captioning literature. In many early works, both image features and word vectors are input to the Long-Short Term Memory (LSTM) language model using the same input matrix, forcing the model to learn a joint embedding for them.

2.4 Approaches to Visual Captioning

Visual captioning techniques include a wide range of methods and models. They can be broadly categorized into two groups: ones generating captions by retrieving from a database [17, 24, 30], and ones using natural language generation techniques to produce captions [16, 35, 36, 65]. While the retrieval-based methods tend to be semantically more accurate as they do not need to learn grammatical rules to generate a caption, the captions they produce

are strictly restricted to the caption database, and thus will not work well on unseen data. In contrast, although generative models can learn to create novel captions and even perform reasonably well on unseen data, they tend to have poorer semantic accuracy and details.

Early work on captioning images presented in [17] was retrieval based, wherein a similarity score between sentences and images is computed and is used to retrieve the best matching caption to an input image. This method relied on few hand-engineered image features and dependency-parsing-based features for sentences. In [24], authors pose image description as a ranking problem involving correctly ranking a set of captions by their relevance to the input image. They argue that the ability to rank captions correctly is a good measure of semantic image understanding, with the added benefit that it is much easier to automatically evaluate such ranked lists than to evaluate generated novel captions. Such retrieval-based system is further enhanced by the use of deep image features and word embeddings in [30].

One of the early models to successfully generate novel image captions was described [35], albeit relying on pre-defined sentence templates to generate captions. In [36], this template based language model is replaced with a n-gram based one learned from large-scale natural language data collected from the web.

Following the successful use of recurrent network–based language models on tasks such as automatic speech recognition [41] and machine translation [3], this approach was quickly adapted to the image captioning literature as well [14, 29, 65]. All these methods consist of two-stage encoder–decoder models, with the encoder being the image feature extraction module and the decoder being the recurrent language model. One major advantage of this approach is that it allows end-to-end training of the entire system. In the case of [29], CNN-based image features and a simple recurrent neural network (RNN) based language model are used as the encoder and the decoder, respectively. The authors also propose techniques to align different parts of a caption to different regions in the image. Similarly in [65], authors also use CNN image features, but an LSTM-based network is used for the language model. In their method, the image features are fed to the LSTM only at the beginning of the recursion, in order to prevent the network from overfitting. Starting from a similar CNN+LSTM based pipeline,[14] proposes a more general framework which can generate captions for both images and videos. In the case of videos, they replace the single image CNN feature with a sequence of conditional random field (CRF) video features and keep the language model configuration identical.

As opposed to such end-to-end learning systems, [16] takes a modular approach. They first train a set of object or concept detectors using multiple

instance learning [40]. Then, a maximum entropy language model takes these detector outputs as input to generate the candidate sentence. This concept detector–based approach has been applied also to video captioning in [51], where the authors train a "visual label" detector on the LSMDC dataset and use it as an input to an LSTM language model.

In [72], instead of using a single image feature vector from a fully connected CNN layer, multiple local image features extracted from a lower layer in the CNN are used. Then an attention mechanism is proposed to choose the right image features to look at while generating different words in the caption. Similar attention mechanism is used in [75], but instead of using the attention model to pick local CNN features, it is used to pick the right semantic concept, from the output of a semantic concept detector.

Attention models extended to temporal domain have been applied to the video captioning task, in order to dynamically choose the right video feature [73]. Alternatively, a recurrent network is used to encode frame-level video features before inputting them to the language model in [64]. Then a standard LSTM language model is used to decode these features into a caption.

2.5 Datasets for Image and Video Captioning

The rapid progress in automatic image and video captioning in the recent years has also been driven by the availability of large-scale datasets to train and test such models on. These captioning datasets have images or videos with one or more associated reference captions. The reference captions can be collected with large-scale human annotation using crowd sourcing tools such as Amazon Mechanical Turk or they can be mined from other related sources.

One of the early datasets for image captioning was Pascal1K [47] consisting of 1000 images five human-annotated captions for each of them. Flickr8k [24] and Flickr30k [76] are relatively much larger datasets, consisting of 8,000 and 30,000 images, respectively. They also have five human-written captions for each image. Currently, the most popular and largest dataset for image captioning is the Microsoft Common Objects in Context (COCO) collection [39] with over 200,000 images and at least five human-written captions per image. There exists also an associated MS-COCO evaluation server, where researchers can upload their captions on the blind test dataset and compare the performance of their system to the state-of-the-art methods on a public leaderboard. Due to its size and availability of a standardized benchmark, all the image captioning experiments reported in this thesis are

conducted on the MS-COCO dataset.

Video captioning datasets are more difficult to collect and we can consequently see both automatic caption mining techniques and manual annotation used to collect them. YouTube corpus [8] consists of 2000 video clips with at least 27 textual descriptions for each video. The M-VAD [60] and MPII-MD [50] datasets used in the first Large Scale Movie Description Challenge (LSMDC), were collected by extracting movie clips and transcribing the associated audio descriptions available in the movie DVDs. In total the LSMDC dataset contains about 100k clips from 202 movies and one reference caption associated with each clip. Creators of this dataset also provide an evaluation server to standardize the testing and comparison of performance on it. More recently released Microsoft Video to Text dataset (MSR-VTT) [71] contains 10,000 short videos with 20 human annotated captions for each of them. This makes it the largest video captioning dataset in terms of video–caption pairs. The MSR-VTT dataset was also used in the recently concluded MSR-VTT video captioning challenge.

We conduct our video captioning experiments on both the LSMDC and MSR-VTT datasets and also report results from our participation of the video captioning challenges associated with these datasets.

Chapter 3

Caption Generation Pipeline: Model and Evaluation

In this chapter, we will examine in detail all the constituent parts of a baseline visual caption generation system. We adapt the model proposed in [65], the basis of the submission which jointly won the 1st Microsoft COCO captioning challenge in 2015, as our baseline model. Although the original model was proposed for generating captions for still images, the same architecture can be used for video captioning, by replacing the image feature extraction module with a video feature extraction module. Thus the discussion presented here is kept generic and specific details of features used for image or video captioning are discussed in Chapter 4.

Then a discussion on the automatic evaluation metrics which are generally used to quantitatively rate the captions generated by the models is presented. The model presented in this chapter acts as the baseline against which we will compare the performance of the architectures and extensions proposed in the rest of the thesis.

3.1 Baseline Architecture

The baseline caption generation model consists of two stages: the visual feature extraction stage followed by a language model. The first stage consists of various techniques to extract descriptors of the visual contents of the input image or video. These descriptors are then represented as one or more vectors of fixed dimension. The language model then uses these feature vectors and generates a suitable caption to describe the image. This pipeline is illustrated in Figure 3.1. In the following subsections, an overview of the different image features and the language model used in the baseline architecture is

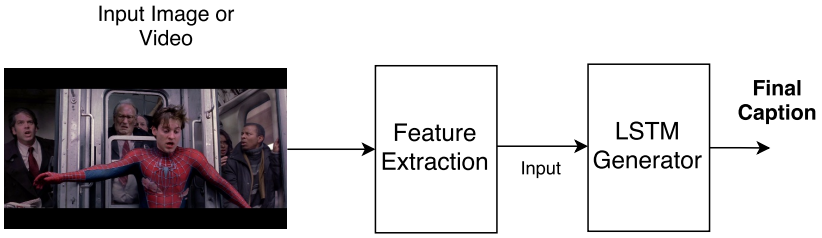


Figure 3.1: A high-level block diagram of the visual captioning pipeline.

presented.

3.1.1 Visual Feature Extraction

Baseline image feature. As discussed in Chapter 2, image features extracted from CNNs pre-trained on ImageNet have become ubiquitous in most image understanding tasks. Therefore, in our baseline captioning model we use features extracted from GoogLeNet [59] as the image feature vector. More details of the feature extraction process are discussed in Chapter 4, but it suffices here to say that the feature vectors are formed by the activations of the *5th Inception module* in GoogLeNet.

Baseline video feature. As a very simple baseline feature vector for videos, we use the same GoogLeNet features as above, but extracted only from a single key frame of the video. We choose the key frame as the frame at the center of the video's time-span. The idea behind using this simple feature vector is to enable the video captioning baseline model to use the same pipeline as for images and to obtain a reasonable baseline against which more sophisticated feature extraction methods can be compared.

3.1.2 Language Model.

The next stage in the pipeline is a conditional language model which takes as input the visual features and generates a caption. The Long-Short Term Memory (LSTM) network [23] architecture has been a popular choice in the literature to model the probability of a sentence S, given an visual feature V, as $P(S|V)$. The following two subsections contain a discussion of the LSTM cell in detail and on the way it is used to build a conditional language model.

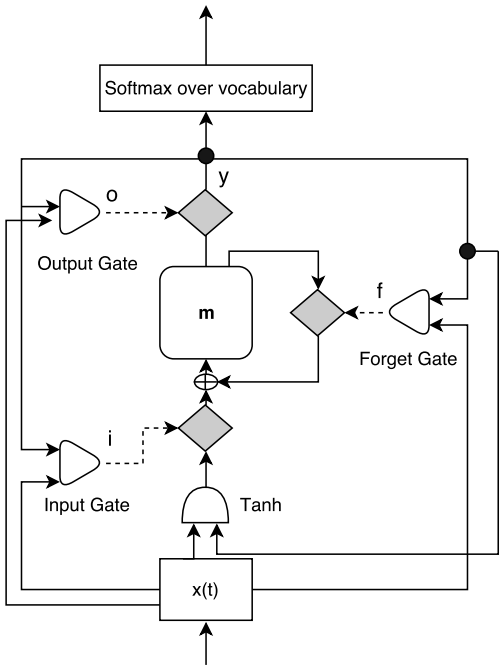


Figure 3.2: A single LSTM cell. Dotted lines to the rhombi indicate multiplications as gate controls and the solid lines depict the data flow. The triangles are sigmoid non-linearities.

3.1.2.1 LSTM Cell

The LSTM model has been chosen as the language model based on two basic requirements the image captioning problem imposes. Firstly, the language model needs to handle sentences of arbitrary length and LSTMs are able to do this by design. Secondly, during the training with gradient descent methods, the error signal and its gradients need to propagate a long way back in time without exploding, and again LSTMs satisfy this criterion.

The block diagram of a single LSTM cell is shown in Figure 3.2. It consists of a memory cell m, whose value at any time step t is influenced by the current vectorial input $x(t)$, the previous output $y(t-1)$ and the previous cell state $m(t-1)$. The update to the memory value m is controlled using the input gate i and the forget gate f. The output is controlled using the output gate o. The gates are implemented with sigmoidal non-linearities $\sigma(\cdot)$ to keep them completely differentiable.

The input and forget gates of the LSTM cells have the ability to preserve the content of the memory cell over long periods, which makes it easier to

learn longer sequences. This process is formalized in the equation system:

$$i(t) = \sigma(W_{ix}x(t-1) + W_{iy}y(t-1)) \tag{3.1}$$
$$o(t) = \sigma(W_{ox}x(t-1) + W_{oy}y(t-1)) \tag{3.2}$$
$$f(t) = \sigma(W_{fx}x(t-1) + W_{fy}y(t-1)) \tag{3.3}$$
$$\begin{aligned} m(t) = f(t) \cdot m(t-1) \; + \\ i(t) \cdot \tanh(W_{mx}x(t) + W_{my}y(t-1)) \end{aligned} \tag{3.4}$$
$$y(t) = o(t) \cdot m(t) \; , \tag{3.5}$$

where $W_{..}$ are the network weights learned during the training phase.

3.1.2.2 Basic LSTM Language Model

The block diagram of the baseline language model is shown in Figure 3.3. The model consists of an LSTM network with a softmax layer at its output. The softmax outputs the probability distribution over the model's vocabulary as:

$$p(w_t|w_{t-1}, \cdots, w_0, V) = \text{softmax}(Dy(t)) \; , \tag{3.6}$$

where D is the decoder matrix which maps the vector $y(t)$, with the same dimensions as the number of LSTM units, to the output vocabulary size, Z.

The visual features V are fed into the LSTM through an embedding matrix W_{ix} at the zeroth time step as the input $x(0)$. We refer to this feature input as the *init* feature since it initializes the hidden state of the LSTM. In the subsequent time steps t, a start symbol followed by the word embeddings for each word in the reference caption (during training) or the previous generated word (during testing) are fed through the same input line, as $x(t)$.

During the training phase, at each time step t, the LSTM is trained to assign the highest probability to the next ground truth word given the current inputs and the hidden state. This is done by maximizing the log likelihood assigned to the training samples by the model. Equivalently, we can minimize the negative log likelihood given as

$$\mathcal{NL}(w_0, \cdots, w_{L-1}|V) = -\sum_{t=0}^{L-1} \log(p(w_t|w_{t-1}, V)) \; , \tag{3.7}$$

where w_{-1} is the start symbol and L is the length of the sentence.

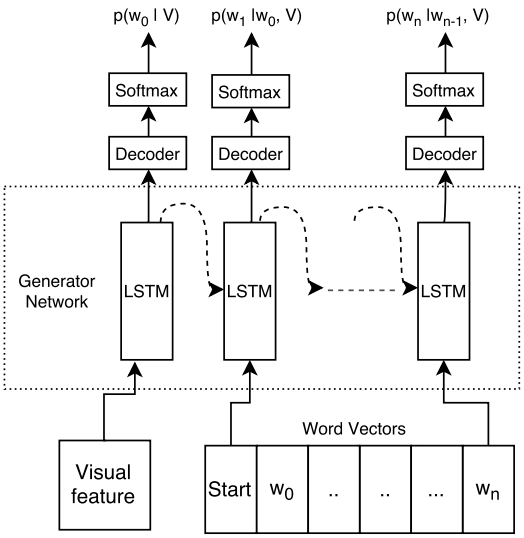


Figure 3.3: The baseline LSTM-based language model with LSTMs unrolled in time.

3.1.3 Training and Regularization

Training the network involves tuning the parameters of the language model in order to minimize the negative log likelihood cost function shown in (3.7). This optimization is achieved by backpropagating the cost through time and adjusting the LSTM parameters using gradient descent. Specifically, stochastic gradient descent with the RMSProp [22] algorithm is used. The training samples are used in random mini batches of sentence–image pairs, and gradient descent is done after accumulating the cost for each mini batch.

All the LSTM parameters, the word embedding vectors and the decoder matrix are learned using this method. The parameters of the feature extraction modules are not trained but are held fixed in order to prevent overfitting. This is necessary as usually the feature extraction modules, e.g. image CNN network, are powerful models pre-trained on large datasets and can easily overfit on the relatively small datasets such as the ones used in captioning.

Dropout is used for regularization during the training of the LSTM language model. Dropout method, as the name suggests, involves randomly dropping the outputs of some neurons in a layer to zero before feeding it to the next layer. This makes the neurons less co-dependent and more robust. As suggested in [78], the dropout is only applied on the input and output of LSTM and not on the recurrent connections. We find that using dropout with a drop probability of 0.5 greatly improves the model generalization. Word embedding vectors and decoder matrix are regularized by adding a

penalty to minimize their l_2 norm.

3.1.4 Test Mode: Beam Search

In the caption generation phase for test images, we do not have a reference caption and need to sample from the distribution $P(S|I)$, to generate the caption. Similar to the training phase, the image feature vector is fed into the LSTM network at time-step $t = 0$ followed by the 'START' symbol. Then, at time-step $t = 1$, the word with the highest probability at the softmax output is the word the model thinks is the most likely first word. We pick it as the generated word and feed the corresponding word vector as the input to the LSTM in the next time step, $t = 2$. This iterative process is repeated until the model produces a period symbol ('.'), which marks the end of the generation process, and we have a complete candidate sentence.

One problem with this approach is that if we only consider the most likely word at each time step we are not guaranteed to get the most likely final sentence. Ideally we should employ an approach akin to dynamic programming and search the entire space of possible sentences. This is intractable as the space of all sentences, even with a finite vocabulary, is infinite as sentences can be arbitrarily long. Even if we limit the sentence length, the search space still grows factorially with the vocabulary size and is still too expensive to search exhaustively. A good approximation is to use beam search, wherein we maintain top b partial sentences at each step. For each of these top-b sentences we consider extensions with top-b words and re-score the partial sentences. Of these b^2 possibilities only the best b extensions are preserved. This process is repeated until all the search beams terminate or maximal allowed sentence length is reached. At the end of this process we have b generated candidate sentences ranked according to the log likelihood assigned by the model.

3.2 Evaluation Metrics

The evaluation of captioning systems is not trivial, due to the non-unique nature of the solution space. An image can be correctly described with a wide variety of captions differing not only in the syntactic structure, but also in the semantic content. We can see an example of this in Figure 3.4, where a sample image from MS-COCO training set is shown with the corresponding ground truth captions. We see that each caption focuses on different aspects of the image, from the *big rock* to *crystal blue water*, but all the captions are equally valid.

C1: a beach with people relaxing on a sunny day.
C2: people are relaxing on the beach where there is a big rock.
C3: a beach with a group of people with surf boards and umberellas.
C4: a group of people enjoy a beach near a lagoon filled with crystal blue water.
C5: a man walking on a beach with his surf board in a case.

Figure 3.4: A sample image from the MS-COCO training set with associated ground truth captions. Here we see a clear case where different captions focus at least partially on different aspects of the image.

A good method of evaluation is to compare the machine-generated caption with multiple human-annotated reference captions. However, we need to note that the reference captions only represent few samples from the space of all valid captions for the image. Having a large number of reference captions makes it more likely that the solution space is better covered by them and thereby leading to more reliable evaluation.

One aspect of the evaluation problem, the syntactic variations in target sentences, is also seen in the well-studied field of machine translation. In this case, a sentence in one language could potentially be translated into multiple valid sentences in the target language. Machine translation still differs from image captioning by the fact that, although these multiple translations can differ syntactically, they tend to have the same semantic content.

Nevertheless, image captioning literature has borrowed three evaluation metrics popular in machine translation, namely BLEU [44], ROUGE-L [38] and METEOR [13]. Another metric popular in image captioning evaluation is the CIDEr metric which was proposed recently in [63], specifically for this task. Next we will discuss each of these four metrics briefly to better understand what exactly they measure.

3.2.1 BLEU

BLEU [44] is a simple metric which scores captions based on the n-gram matches between the candidate and the reference captions. First, occurrence counts of different n-grams in the candidate sentence are counted and clipped to their maximum value in any single reference sentence, and then

accumulated. Next, a modified precision score is computed by dividing this accumulated score by the total number of n-grams in the candidate. This process is repeated for different n-grams, yielding modified precision scores p_n. The final BLEU score is given by

$$\text{BLEU-n} = BP \cdot \exp(\sum_{n=1}^{N} w_n \log(p_n)) \ , \tag{3.8}$$

where BP is the brevity penalty applied in order to penalize short candidate sentences. This additional term is required since, if we only use precision, degenerate candidates such as the ones containing just single words will always score better than longer sentences.

3.2.2 ROUGE-L

ROUGE [38] metrics were proposed for evaluating text summaries. The version used in image captioning evaluation is ROUGE-L, a metric based on recall and precision scores of the longest common subsequences (LCS) between the reference and candidate sentences:

$$R_{lcs} = \frac{LCS(Cand, Ref)}{Reference\ length} \ , \tag{3.9}$$

$$P_{lcs} = \frac{LCS(Cand, Ref)}{Candidate\ length} \ , \tag{3.10}$$

where R_{lcs} and P_{lcs} are recall and precision metrics, $LCS(Cand, Ref)$ is the longest common subsequence between the candidate $Cand$ and reference Ref.

The metric looks for common sub-sequences by looking for words which appear in the same order in both the reference and candidate captions. Note that these words need not be consecutive, just in-sequence. Finally, the ROUGE-L metric is computed as the F_β score with $\beta = P_{lcs}/R_{lcs}$:

$$\text{ROUGE-L} = F_\beta = \frac{(1+\beta^2) R_{ics} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \ . \tag{3.11}$$

3.2.3 METEOR

In order to compute the METEOR [13] metric, the candidate and reference sentences are first aligned, wherein each word in the candidate sentence is matched to at most one word in the reference. When matching words between the candidate and the reference, apart from the exact match, WordNet

synonyms, stemmed token matches and paraphrase matches are also considered, in that order. The alignment is done so as to minimize the number of chunks, i.e. the group of words which are consecutive and in order in both the sentences. Once the two sentences are aligned, weighted precision and recall are computed on the matched words, with different weights being applied to different kinds of matches. The final METEOR score is computed as the product of a penalty term to penalize the number of chunks in the alignment and the F-score based on the weighted precision and recall:

$$Pen = \gamma \cdot \left(\frac{ch}{m}\right)^{\theta} , \tag{3.12}$$

$$\text{METEOR} = (1 - Pen)\frac{P_m R_m}{\alpha P_m + (1-\alpha) R_m} , \tag{3.13}$$

where R_m and P_m are the recall and precision metrics, ch is the number of chunks the alignment has, m is the length of the candidate and α, γ and θ are hyper-parameters tuned to maximize the correlation of the metric with human judgment. When there are multiple references, the maximum METEOR score between the candidate and any reference is taken.

3.2.4 CIDEr

CIDEr [63] metric aims to measure how well the candidate caption matches with the consensus formed by the multiple reference captions. For this purpose, each candidate caption and reference sentence is represented using term frequency inverse document frequency vectors (TF-IDF). TF represents the consensus, by considering frequently occurring terms in the reference captions, and IDF helps down-weight common words which occur across captions for many different images.

CIDEr_n metric is computed by averaging the cosine similarity between the TF-IDF vectors of the candidate caption and all reference captions. Here n is the n-gram size considering which TF-IDF vector was formed. Final CIDEr metric is the mean of the four CIDEr_n metrics, with $n = 1, 2, 3, 4$.

Few modifications were made to this original CIDEr metric to prevent gaming, i.e. producing captions which could achieve higher CIDEr scores but fail badly in human judgments. This is referred to as CIDEr-D and is the version widely used in image captioning.

3.2.5 Reliability of the Automatic Metrics

In summary, all the four metrics discussed here evaluate the suitability of a caption to the visual input, by comparing how well the candidate caption

matches the reference captions. They perform better with the increasing number of reference captions as was found in [63]. The same study also found that CIDEr and METEOR have the highest correlations to human judgment, followed by ROUGE-L and finally BLEU-4.

All the four metrics were used to rank submissions in the first MS-COCO image captioning challenge. The competition also collected human judgments to evaluate the submissions. On comparing the two, it was found that the rankings produced by METEOR and CIDEr were the ones most correlated with the ranking of submissions as per human judgment [11], reaffirming the observations made in [63]. Similar trend was also seen in the first Large-Scale Movie Description Challenge (LSMDC) 2015. The LSMDC 2015 used both the automatic metrics and the human judgments to rank the submitted captions to the test videos. Even here, the ranking of the submissions produced by the automatic metrics poorly correlated with the rankings produced by human judgments.

Hence in Chapter 6, when evaluating the visual features and language model architectures proposed in this thesis, most importance will be given to METEOR and CIDEr metrics. We will also present results from human judgments of the video captioning models presented in this thesis, obtained through participation in the LSMDC 2015 and the MSR-VTT challenges.

Chapter 4

Enhancing Visual Features

Finding good feature vector representations for the input images and videos is a very important task for successful design of a captioning system. Such a feature representation should be compact, but also able to encode all the information relevant for the task. For the image captioning task, the feature vector should capture all the objects in the image, their most essential properties such as color, their absolute position and relative location to each other, along with the type of the scene these objects are located in. In case of video captioning, apart from all the above mentioned information, the feature vector should also encode sufficient temporal information to enable recognizing actions, order of events, etc.

In this chapter, we will study different visual features for both images and videos in order to improve the performance of the captioning system over the baseline presented in Chapter 3. The chapter is divided into two main sections, with one discussing the visual features used to represent images and the other describing the features used to encode videos.

4.1 Image Feature Extraction

Activation values extracted from the deep Convolutional Neural Network (CNN) layers are the primary features used to represent images in the captioning models presented in this thesis. The CNN features are able to encode a rich variety of information, including scene context, object type, etc., as seen from its performance in the baseline model. However, this representation is still very dense and probably (as seen later in experiments) inefficient for the language model to be able to extract the information it needs to generate correct captions. It is also unclear to what extent these features encode multiple objects and object locations, since they are trained on the

ImageNet task involving recognizing a single object class.

Thus, in a bid to improve the performance over the baseline model, language model is provided with additional features which explicitly encode presence of objects, scene types and object location. To achieve this, explicit object detectors and scene detectors are trained based on the CNN features. Additionally, features encoding object localization are constructed based on outputs from Faster Region-based Convolutional Neural Network (R-CNN) [48]. In the following subsections we will discuss the exact details of the processes used to extract all of the above features from input images.

4.1.1 Convolutional Neural Networks

Convolutional Neural Networks have in the recent years become the most widely used models for practically all tasks related to image classification and understanding. It is shown in [15] and [54] that activations of the fully-connected layers of a CNN trained for image classification task act as a general feature representation of the image and can be successfully used to solve other tasks as well. In line with this, image features are extracted here from different CNN architectures pre-trained on two large datasets namely, ImageNet [12] and MIT Places [79], originally aimed for object and scene classification, respectively.

The CNNs used here are based on the widely used GoogLeNet [59] and VGG [58] architectures. Both of these architectures achieved good results in the ILSVRC 2014 object classification challenge, finishing first and second, respectively. In this work, the GoogLeNet features are used both as a direct input to the language model and to train the place and object detector modules, while the VGG net features are only used for the latter.

4.1.1.1 GoogLeNet

The main idea behind the GoogLeNet [59] architecture is to use small dense structures like 1×1, 3×3 convolutions to mimic a large sparse layer. For this purpose they utilize the *Inception* modules consisting of 1×1, 3×3 and 5×5 convolutions and maximum pooling layers. This network achieved the top-5 error rate of 6.67% in the 1000 class ILSVRC 2014 Classification Challenge, finishing first in the competition.

We use two different versions of the GoogLeNet features in our experiments. Features from GoogLeNet trained on ImageNet [12] dataset are used as direct input to our language model (referred to as "*gCNN*" in the rest of the text), whereas features from the GoogLeNet trained on MIT Places [79] data (referred to as "*pCNN*" in the rest of the text) are used to construct

scene recognition features which are used as auxiliary inputs to our language model.

To extract *gCNN* features, we use the activations from the *5th Inception module*, having the dimensionality of 1024. We augment these features with the reverse spatial pyramid pooling proposed in [19] with two scale levels. The first scale is just the full image rescaled to the size of 224×224. The second level consists of a 3×3 grid of overlapping patches of size 128×128 with stride of 64, and horizontal flipping. The activations of these regions are then reduced to a single 1024 dimensional feature vector by using average pooling or by just using the central crop. Finally, the activations of the two scales are concatenated resulting in 2048-dimensional features. Note that due to two different pooling methods on the second scale, we obtain two somewhat different feature vectors of 2048 dimensions from the same network. Our final *gCNN* feature vector of size 4096 dimensions is obtained by concatenating these two feature vectors. This is also the feature vector used as the input to our baseline image captioning model.

The same procedure described above for the ImageNet trained GoogLeNet has also been followed with the Places data trained GoogLeNet, with the exception that instead of the Inception module, the *3rd classification branch* has been used as the activation layer where the feature vectors have been extracted. In this case, in addition to the mean and center pooling in the second scale of reverse spatial pyramid, we also use maximum pooling, and thus obtain three different features with the dimensionality of 2048 each. Note that the term *pCNN* refers collectively to this set of three features.

4.1.1.2 VGG Network

VGG network was introduced in [58], where the authors study the effect of depth on the performance of convolutional networks. The salient feature of this architecture is the exclusive use of small 3×3 and 2×2 convolutional filters throughout the network. This helps in keeping the number of parameters small even with the increased depth. This network achieves the top-5 error rate of 7.3% in the 1000 class ILSVRC 2014 Classification Challenge, finishing second in the competition.

Two variants of the VGG net, namely the 16- and 19-layered ones from [58], are used in the experiments reported here. From both the variants, we extract the activations of the network on the second fully-connected 4096-dimensional *fc7* layer for the given input images whose aspect ratio is distorted to a square. Ten regions, as suggested in [34], are extracted from all images and average pooling of the region-wise features are used to generate the final features.

4.1.2 Object Detectors

As mentioned before, the CNN features are augmented with explicit object detector features where each dimension represents the presence or absence of one of the 80 object categories defined in the COCO dataset [39]. These 80 categories consists of common object types such as *person*, *car*, *bus*, *dog*, *cat*, *table*, *chair*, *pizza*, *banana*, *laptop*, etc.

To construct the explicit object detector features, 80 separate SVM classifiers [10] are trained on the COCO 2014 [39] training set to detect each of these 80 object categories. Image features extracted using the previously described five CNN-based ImageNet-trained VGG and GoogLeNet features are used as input to the 80 object detector SVMs. In particular, we here utilized linear SVMs with homogeneous kernel maps [62] of second order to approximate the intersection kernel. Furthermore, we used two rounds of hard negative mining [37] and sampled 5 000 negative examples on each round.

For each image we thus have 15 SVM outputs for each class (five features times initial and two hard negative trained models) that we combine with simple arithmetic mean in the late fusion stage. The 80 fused output values, one for each object category, are then concatenated to form a class membership vector for each image. These vectors we optionally use as inputs to the LSTM network and we denote it as "*SVM80*" in the rest of the thesis.

4.1.3 Scene Detectors

In order to provide the language model with explicit information on the visual environment or the scene type of the images, we used the SUN Scene Categorization Benchmark database [70] and [69] to create a bank of visual detectors specialized for scene recognition. The version of the database we used contains 108,756 images associated with one of 397 scene categories. The 397 categories include three major classes — namely indoor, outdoor-natural and outdoor-man-made — and common scene types including *kitchen*, *living-room*, *shower*, *tennis-court*, *courtroom*, *beach*, *dock*, *airfield*, *dam* etc.

We extracted both ImageNet data trained and MIT Places data trained GoogLeNet CNN features, as described in the previous section, for the images in the SUN database. We used features of all the images (not only the training split) for training Radial Basis Function (RBF) Support Vector Machines (SVMs) with the LIBSVM software [7]. As we had three slightly different versions of each of the feature types, we obtained the total of six SVM detectors for each scene category.

We applied each of the detectors to the images of the COCO dataset and

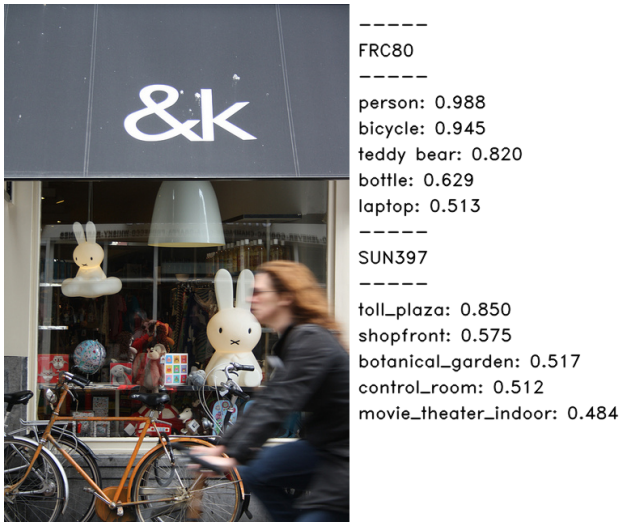


Figure 4.1: Visualization of top five detected objects (FRC80) and scene types (SUN397) along with the associated confidence scores for an image from the MS-COCO validation set.

used the simple arithmetic mean for the late fusion of the detector outputs. The concatenation of the fused category-wise detector outputs results in 397-dimensional feature vectors for the respective images. These feature vectors are referred to as "*SUN397*" in the rest of the thesis. Figure 4.1 shows the top five detected scene categories using the *SUN397* features for an image from the COCO validation set.

4.1.4 Spatial Map Encoding

Another important source of information relevant to generating captions is the relative location of the objects in the image. Knowing the locations of the objects helps to infer their roles and actions in the scene and also to choose the right adjectives to describe their positions.

For this purpose we use an object detector network, specifically the Faster Region-based Convolutional Neural Network (R-CNN) proposed in [48], to detect the multiple objects in the input image and to predict their bounding boxes.

We project the object bounding boxes onto a non-overlapping $m \times n$ grid over the image. Each of the 80 object categories, c, maintains its own grid F_c. Each grid cell, $F_c(i, j)$, accumulates the intersection over union (IoU) value of any overlapping bounding box of that object category. This IoU value is also scaled with detection confidence score generated by the Faster R-CNN

for that bounding box. Thus, we get 80 spatial maps of size $m \times n$. These representations are then concatenated to produce an $m \times n \times 80$ -dimensional feature vector. The value of the feature vector component corresponding to the class c and the grid cell position (i, j) can be computed as

$$F_c(i,j) = \sum_{b_k \in BB(c)} p(b_k) \frac{A(b_k \cap G(i,j))}{A(b_k \cup G(i,j))} , \tag{4.1}$$

where $A(\cdot)$ is the pixel area, $BB(c)$ are the bounding boxes detected for objects of class c, $p(b_k)$ is the confidence assigned by the detector to box b_k and $G(i, j)$ is the grid cell at position (i, j). We abbreviate these features as "$m\times n$IoU" in the result tables.

We also experiment with replacing the bounding box with a Gaussian whose mean is at the center of the bounding box and standard deviation is the length of the box diagonal. Then, instead of the IoUs, each grid cell accumulates the integrals of the Gaussians in their overlapping region as

$$F_c(i,j) = \sum_{b_k \in BB(c)} p(b_k) \iint_{b_k \cap G(i,j)} N(x, y, \text{center}(b_k), \text{diag}(b_k)) dx dy , \tag{4.2}$$

where $BB(c)$ is the set containing bounding box object proposals for category c, $p(b_k)$ is the confidence assigned by the detector to the bounding box proposal b_k, $G(i, j)$ is the grid cell at position (i, j) and $N(x, y, \mu, \sigma)$ are Gaussians in x and y variables of given mean μ and standard deviation σ. The double integral in the above equation is done computed the area of intersection between bounding box, b_k and grid cell $G(i, j)$, $b_k \cap G(i, j)$. We abbreviate these features as "$m\times n$Gauss" in the result tables.

As an alternative to the $m \times n$ non-overlapping grid, we have used also "$m+n$ partitioning" of the images. By this we mean that the images are split into m horizontal and n vertical regions, each of equal width and equal height, respectively. The localization information provided by this representation is not as accurate as that of the $m \times n$ grid model. Instead, it can be argued to be more robust against variations in the relative placements of the objects. The calculation of these features is analogous to that of the grid-based ones, only the definition of the grid cells $G(i, j)$ is different. We abbreviate these features as "$m+n$Gauss" in the figures and result tables.

By setting the grid size parameters $m = n = 1$ we lose all localization information and get plain Faster R-CNN based object detector feature vectors, similar to the SVM80 features described in the previous section. These vectors are referred to as "FRC80" in the rest of this text. An illustration of the top five objects detected using the *FRC80* features for an image from the COCO validation set is shown in Figure 4.1.

We have to note here that it is computationally relatively expensive to extract these features, as running the Faster-RCNN network on the input image which, despite the moniker, is still much slower than running a single CNN such as the GoogLeNet on the input image.

4.2 Video Feature Extraction

We use two different paradigms for video feature extraction. The first one is to treat the video as just a sequence of 2-D static images and use CNNs trained on ImageNet [12] to extract static image features from these frames. The second approach is to treat the video as 3-D data, consisting of a sequence of video segments, and use methods which also consider the variations along the time dimension for feature extraction.

In both the approaches above, pooling techniques are used to combine multiple frame or segment level features into one video-level feature vector. Then number of feature vectors can also vary depending on the length of the video. Since our language model cannot take arbitrary number of feature of vectors as input, we need to further compress these multiple feature vectors into a single video feature vector. We use mean pooling, which is just taking the average of these feature vectors, to reduce them to a single vector.

4.2.1 Keyframe and Multi-Frame Features

In this subsection we will discuss the frame-level features used in the video captioning model proposed in this thesis. The frame-level video features are extracted by treating the video as a bag of static images, and applying the same feature extraction techniques used for image captioning to these video frames. The key idea is to capture details of the scenes shown in the video, objects present in it and their attributes. If the clip is very short and consists of only a single shot, then it might be sufficient to extract these features from a single keyframe extracted from the center of the video. However, if the video clip is long, one will need to extract the frame-level features from multiple frames, to sufficiently summarize the content in the video.

In order to keep the computational time reasonable, all the feature extraction methods discussed before for images are not used on videos. Instead, only the gCNN features and the explicit object recognition features (*SVM80*) are extracted from the video frames. Computational efficiency is also improved by sampling only the keyframe in case of LSMDC dataset and one frame every second in MSR-VTT dataset for feature extraction. The feature

Figure 4.2: A visualization of the improved dense trajectory features for a video from the MSR-VTT training set.

extraction procedure for both gCNN and SVM80 features remain the same as described before for images in sections 4.1.1.1 and 4.1.2.

4.2.2 Segment-Level Features

Although the frame-level features do well in capturing the overall scene context of the videos, they fail to recognize actions and other motion-related events. This is because lot of action in videos involve very local motion patterns, which is hard to capture when looking at frames individually. Such actions can occur even without any whole objects moving, and thus causing the CNN feature extractors to not notice hardly any change.

The motion information is, however, vital for the video captioning task, as describing the salient actions occurring in the video correctly is an important part of the caption. Thus we need video feature extraction method, which can effectively encode information about local motion patterns existing in the video into a fixed-size feature vector. To meet this requirement, two feature extraction methods which treat videos as a series of video segments and operate on these video segments to extract motion information are utilized. These are dense trajectories [66, 67] and 3-D CNN network based C3D features [61] and we will discuss them in detail in following subsections.

4.2.2.1 Dense Trajectory Features

Dense trajectory video features have been one of the best-performing features on various video analysis tasks involving action recognition, despite being

hand-engineered features without any learning from the data. The technique involves sampling interest points from an initial frame in the video and tracking these interest points across time. The location of the interest points which can be tracked reliably are concatenated to obtain several trajectories in the video. Local descriptors are then extracted from patches around the points in the trajectory which serve as features representing the shapes of moving objects. In this thesis, the same four local descriptors as described in [66, 67] are utilized— namely histogram of oriented gradients (HOG), histograms of optical flow (HOF) and motion boundary descriptors in horizontal and vertical directions (MBHx & MBHy). Videos can then be represented based on the distribution of the trajectories they contain. This can be accomplished by using vector encoding methods such as the bag of words histogram or Fisher vector encoding [46] to compress the arbitrary number of trajectory features extracted from a video into a single vector of fixed dimensions.

Both the standard [66] (DT) and the improved versions [67] (IDT) of the dense trajectory features are utilized in this thesis. In both versions, the trajectories and their descriptors are first extracted from the entire video after limiting the trajectories to be a maximum of 15 frames long. A visualizations of trajectories extracted using the IDT method from a video in the MSR-VTT training set is shown in Figure 4.2. We can see in this example that most of the trajectories are concentrated around the man performing an action.

Each of the five types of features extracted, trajectory co-ordinates and the four descriptors, are separately encoded into fixed-size vectors using the bag-of-features encoding with a codebook of 1000 vectors. In the bag-of-features encoding, first a fixed codebook containing representative sample features from the training set is chosen. In our case, the codebook is obtained using k-means clustering on random 250k trajectory samples from the training set, with k=1000. Then, each feature vector from the set of features of a video is assigned to its nearest codebook vector. These assignment counts are accumulated for each codebook entry to give us a histogram for each video, with the histogram having 1000 bins. Finally, concatenating the vector encodings of each of the descriptors we get a video feature vector of 5000 dimensions.

4.2.2.2 3D Convolutional Network

As an alternative to the hand crafted dense trajectory features, video-segment features are extracted using a deep neural network based on 3-D convolutions. Specifically, the C3D [61] network, pre-trained on the Sports-1M dataset, is used.

Inspired by the success of deep 2-D convolutional neural networks as

image feature extractors, the C3D model attempts to employ similar deep networks to learn video representations. Since videos have an additional temporal dimension, the 2-D convolutions used in image CNNs are replaced with 3-dimensional convolutional filters in C3D. Influenced by the use of small 3×3 kernels in the VGG [58] network, C3D uses only $3 \times 3 \times 3$ convolutional filters. To prevent too quick loss of temporal information quickly, the pooling operations are also kept to small windows of size $2 \times 2 \times 2$.

The C3D network used in this thesis is pre-trained on the large Sports-1M dataset, for the task of classifying the video into one among 487 classes of sports-related actions. To extract features using C3D, the input videos are first cut into non-overlapping video segments of 16 frames long. Then each segment is input into the C3D network and the activations from the *fc6* and *fc7* layer of the network are extracted as the segment-level features. This results in a set of 4096-dimensional feature vectors which represent the input video and are input to the language model after pooling.

Chapter 5

Enhancing Language Model

In this chapter we will look at several extensions to the baseline language model. The first extension is to add an additional feature input channel to the language model aimed at helping us effectively utilize the multiple new image features proposed in Chapter 4. We then discuss an adaptation of the residual connections, recently proposed for CNNs, to the LSTM language models. This allows us to effectively train deeper LSTM language models. Next we look at utilizing a hierarchical factorized decoder at the language model output, in a bid to produce richer captions. Finally, a few ensembling techniques for combining multiple language models are also presented.

5.1 Additional Feature Inputs

The language model we use should be capable of simultaneously utilizing different kinds of visual features presented in Chapter 4. However, the baseline language model, presented in Chapter 3 has only one input channel, shared between the word vectors and image features. Hence, in the baseline model the visual features are input to the network only in the zeroth round of iteration. We refer to this input channel as the *init* input to the LSTM network. This technique was also proposed in [65] as a solution to prevent overfitting. Therefore, the only way the baseline language model can utilize multiple visual features is if they are fused into a single vector before presenting to the language model. In our experiments, we have unfortunately found that performing simple feature fusion, such as concatenating the two feature vectors, and using it as an input in the baseline language model leads to inferior performance.

To address this issue, another data input channel is introduced to the language model proposed in this thesis. Furthermore, utilizing the new input

channel, the visual features are made available through the whole inference process. This requires adding a new input path to the LSTM cell which we refer to as the *persist* input. This is illustrated in figure 5.1. Note that the *persist* input plays the same role as $x(t)$ in equations (3.1)–(3.4), but it has its own set of input weights. For brevity, we will not repeat them here. Accordingly, the proposed framework computes the distribution

$$p(w_t|w_{t-1}, \cdots, w_0, I, P) = \text{softmax}(Dy(t)) \ , \tag{5.1}$$

where I and P represent the *init* and *persist* features respectively. Training is again done by minimizing the negative log likelihood cost function as in the baseline model:

$$\mathcal{NL}(w_0, \cdots, w_{L-1}|I, P) = -\sum_{t=0}^{L-1} \log(p(w_t|w_{t-1}, I, P)) \ . \tag{5.2}$$

Having separate input matrices for *init* and *persist* features allows the model to learn different functions from the word embeddings and the visual features and in-turn makes the language model more powerful. We can also now input different visual features in the *init* and *persist* paths thereby allowing the model to learn simultaneously from two complimentary sources.

5.2 Deeper Models Using Residual Connections

Another extension we experiment with to improve the performance of our language model is to add depth to the LSTM network used in it. Here, only the first LSTM layer receives the feature inputs directly and the higher LSTM layers take their input $x(t)$ from the previous layer in the network. The recurrent connections from a network's outputs back to its inputs exist only within an LSTM layer and not across the layers. Softmax is applied at the output of the last LSTM layer only.

Residual connections were recently proposed in [21] to be used in CNNs. The method consists of adding a fixed identity connection from the output of a lower layer, $f_1(x)$, to the output the layer above it, $f_2(x)$. This alters the output after two layers from $f_2(f_1(x))$ to $f_1(x) + f_2(f_1(x))$. Now the second layer only needs to learn a residual function to push the output of the first layer towards the desired output. If the first layer is already performing well, this makes the task of the second layer much easier. This seemingly simple technique allowed to train much deeper CNNs (almost three times of the

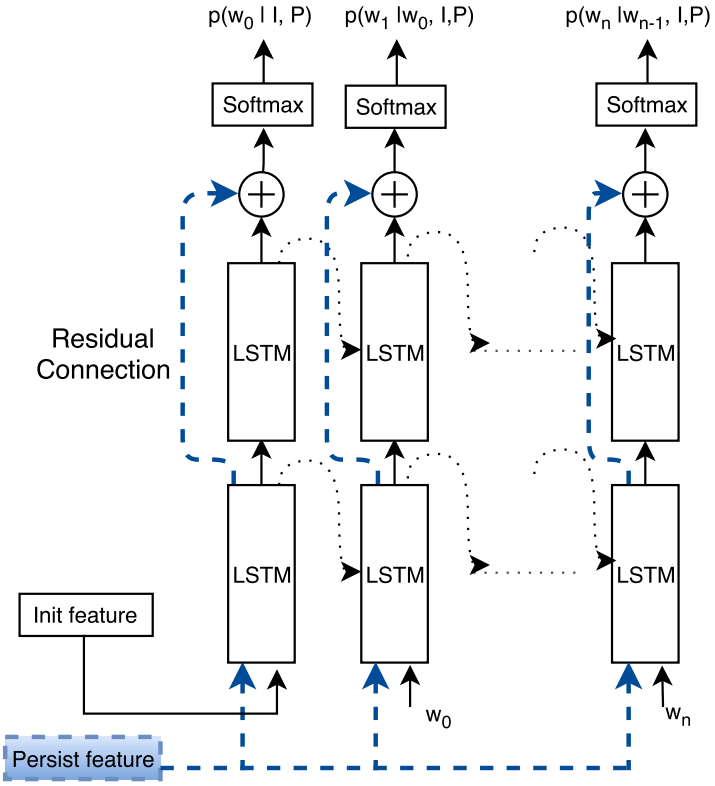


Figure 5.1: The proposed language model architecture. The dashed blue lines highlight the changes proposed over the baseline. A two-layer LSTM with residual connections is shown here.

GoogLeNet) and achieve state-of-the-art performance on ImageNet dataset in the image classification task [21].

Inspired by this success, we adapt the method to our LSTM language model and add residual connections between the LSTM layers. These residual connections, shown in Figure 5.1, greatly improve the training convergence speed. We have also found out that the use of residual connections produces significantly lower values for both the training cost function and the model perplexity, as will be presented in Section 6.2.2.3.

5.3 Hierarchical Decoder

Analyzing the erroneous outputs from the baseline language model, one category of mistakes we often see are where the model misses the fine-grain classification between two closely related objects. For example, we have no-

ticed that a person's gender is often described wrongly. Similar uncertainty is also seen in telling apart various fruits and vegetables. This happens because these objects often occur in similar language contexts and hence the context cannot help differentiate them. To correctly identify them, the LSTM generator needs to learn really fine-grained object classification.

In this thesis, it is hypothesized that the above problem is exacerbated by the language model essentially having to do a Z-way classification at every time-step, where Z is the size of the vocabulary. When the size of the vocabulary is large, e.g. on the COCO dataset the vocabulary contains 8790 words, this Z-way classification is a hard problem. Additional complexity arises because the errors made in the Z-way classification at any time-step affects all future predictions made during the generation of that caption.

We attempt to address this problem by splitting the Z-way classification into two smaller hierarchical classification problems. To do this, the language model vocabulary is first split into Z^c classes and each word in the vocabulary is assigned to one class. Then the conditional probability of a word given the context can be factorized into two parts as follows:

$$P(w_t|I, P, W^{t-1}) = P(c_t|I, P, W^{t-1})P(w_t|c_t, I, P, W^{t-1}) \,, \tag{5.3}$$

where c_t is the class of word w_t and W^{t-1} is sequence of words seen up to time $t-1$, $(w_0 \cdots w_{t-1})$.

This allows us to split the Z-way softmax at the LSTM output into two parts, a Z^c-way softmax to predict the correct class and a Z^c_w-way softmax to predict the correct word within this class. Here Z^c_w is the number of words within the class c. This hierarchy allows the language model to learn separate decoders for each class, possibly allowing it to learn more fine-grained classification. The hierarchical structure is also more amenable to adding new words to the vocabulary. Adding a new word to a specific class already transfers the knowledge the model has about the class to the word, and the model only needs to learn the probability distribution of the new word within the class.

5.3.1 Clustering Words to Classes

The first step in implementing the hierarchical structure in the decoder is to cluster words into different classes. There are two kinds of approaches to do this clustering in the literature. The first group of methods use knowledge bases such as WordNet to find similar words and group them together. The second category of methods are completely data-driven and use just the training data statistics to build the clusters. Here we will look at two differ-

ent methods relying solely on the COCO training data to cluster words into classes.

Brown Clustering

A popular data-driven clustering algorithm is *Brown clustering* proposed in [6]. It is a hierarchical agglomerative clustering algorithm which produces a hierarchical tree-like clusters of all the words in the vocabulary. To begin with, all the words in the vocabulary are assigned to individual leaf nodes. Then, starting from the leaf nodes, two nodes whose merging would reduce the clustering cost function the most are combined into a single node. The clustering cost function is simply the perplexity assigned by a class based bigram model to the training corpus. In our case the training corpus consists of all the reference captions in the training set. The clustering cost can be written as a function of the given cluster assignment, $C(.)$, as

$$Quality(C) = \frac{1}{n} \log \prod_{i=1}^{n} P(C(w_i)|C(w_{i-1}))P(w|C(w_i)) \ , \tag{5.4}$$

where $C(.)$ is the mapping which assigns words w to their cluster $C(w)$.

The process is terminated once the number of nodes have been reduced to the desired number of clusters, Z^c. Although this method produces hierarchical clusters, one can ignore the hierarchy and take all the classes in the final output as independent classes in the language model.

K-means Clustering

K-means is a simple yet very effective clustering algorithm widely used on high-dimensional vectorial data. The algorithm tries to partition the set of input data vectors into K clusters such that each data point belongs to the cluster whose mean is closest to it. Usually, the Euclidean distance is used to measure distance between data points. One cannot, however, directly use the K-means method to cluster words as one cannot measure the distance between words. Instead, the word-embeddings which are learned in the language model can be used to represent words as vectors in d-dimensional space. With this modification, the Euclidean distance between word embeddings can be used as a measure of distance between the words and thus the K-means algorithm can be utilized to find into Z^c cluster centers in the word embedding space, by setting $K = Z^c$. Once Z^c cluster centers are obtained, each word is assigned to its closest center and this partitioning is used in the language model.

5.3.2 Factorizing LSTM Decoder Output

In order to implement the hierarchical decoder in our LSTM language model, we need to split the decoder matrix, D into a set of $Z^c + 1$ smaller matrices. This set contains one D_{cls} matrix of size $h_{sz} \times Z^c$ which is used to compute the class probability. Here, h_{sz} is the hidden size of the LSTM layer in the language model. It also has K class specific matrices $D^1, D^2 \cdots D^{Z^c}$, each of the size $h_{sz} \times Z^c_w$. Here Z^c_w is the number of words in the class c. Consequently, computation performed to predict the word at time-step t has two stages, first to predict the right class of the word and then using the class specific decoder matrix to predict the word within the class:

$$P(c^k_t|I, P, W^{t-1}) = \text{softmax}(D_{cls}y(t)) \tag{5.5}$$

$$c_t = \arg\max_k \left(P(c^k_t|I, P, w^{t-1})\right) \tag{5.6}$$

$$P(w_t|I, P, w^{t-1}) = P(c_t|I, P, w^{t-1}) \cdot \text{softmax}(D^{c_t}y(t)) \tag{5.7}$$

In theory this model should be faster since we only need to compute the two smaller matrix multiplications instead of the one large one. However, due to bottlenecks in our implementation, this hierarchical decoder runs slower than the single-stage decoder.

5.4 Ensembling Techniques

Using the many different image features and the LSTM language model architectures we have discussed before, we can train a set of different language models. When examining the pool of captions generated by such models for a set of images, we have found out that different models tend to generate the best captions for different kinds of images and videos. This indicates that ensembling these different generative models could be a good idea. If one could evaluate the suitability of a given caption for a given image, one could possibly pick out the best candidate from the pool and achieve better results than with any single model. In this section we will examine two methods of evaluating the suitability of a caption to the visual input and thus effectively ensembling multiple language models.

Concretely, given an input image or video feature, V, and a set of p candidate captions, $C_p = \{S_1, S_2, \cdots, S_p\}$, generated by m language models, $LM = \{lm_1, lm_2, \cdots, lm_m\}$, we wish to find the evaluation function $E(S|V)$, such that $\arg\max_{S_i} E(S_i|V)$ is the most suitable caption for V in the set C_p. The value of $p \leq (b \times m)$, with $p < (b \times m)$ when same captions are generated by multiple models. Note that this evaluation function is different from using

evaluation metrics to evaluate a captioning system, as here we are assessing the captions without using the ground truth.

The first method relies on the caption-generating language models themselves to evaluate the candidates, while the second method involves training a separate evaluator model to measure the similarity between the candidate captions and the visual input. We will examine them in detail in the following subsections.

5.4.1 Combining Models with Mutual Evaluation

In this technique, we utilize the same language models which generated the captions to also evaluate the captions. All the LSTM language models presented so far learn the conditional probability distribution of the caption given the visual input, $P(S|V)$. We utilize this probability, $P(S|V)$, as a measure of the goodness of the caption w.r.t the input, and then use it to rank the candidate captions. Considering only the probability assigned to a caption S_i by the language model which generated it, we could pick the caption with the highest probability score as the best caption. We refer to this method as "Self-Eval" in rest of the thesis.

Alternatively, one could also get the candidate captions evaluated by all the other models in the ensemble. Thus one gets m scores per caption, which can then be used to pick the best candidate either using the "max-mean" method or the "max-max" method. In the "max-mean" case, the candidate with the highest mean of probability scores is chosen as the best caption. In the case of the "max-max" method, the candidate with the highest maximum among the m probability scores is chosen as the best candidate:

$$\text{max-mean:} \quad S_{best} = \arg\max_{S_j} \left(\frac{1}{m} \sum_{i=1}^{m} P_{lm_i}(S_j|V) \right) , \tag{5.8}$$

$$\text{max-max:} \quad S_{best} = \arg\max_{S_j} \left(\max_{i} P_{lm_i}(S_j|V) \right) , \tag{5.9}$$

where P_{lm_i} is the probability distribution learned by model lm_i.

Since in this method models are evaluating each others' sentences, we refer to it as "Mutual-Eval". This method is also similar to the peer-review model used in academic publishing, where researchers with expertise in related fields evaluate each others' work. It works best when all the models in the ensemble are equally competitive, with expertise in slightly different sub-domains of the dataset.

5.4.2 CNN Evaluator

The language models we described for the evaluation of candidate captions in the previous subsection are trained generatively. Thus they need to learn to model both the semantic and syntactic structures of a sentence and their relation to the input image. In our experiments, we have noticed that these language models indeed are very effective at learning syntactic structures of sentences, and grammatical mistakes are rare in the generated captions. Most of the errors in the generated captions are of semantic nature, wherein the models tend to get the objects or the relations between the objects wrong. This could be because similar syntactic structures repeat across a large number of reference captions in the training dataset, but similar semantic relations are found only in smaller parts of the dataset. This indicates that the evaluator function should mainly focus on evaluating the semantic correctness of the candidate captions, and not worry about nitty-gritties of syntactic correctness. Thus, a generatively trained model will be suboptimal at this task and discriminative training would be more suitable.

A solution to address this is to discriminatively train a new model whose task is to pick out the best candidate from the candidate set, given an input image or video. We refer to this as an evaluator network. The network takes as input one visual feature vector and an input sentence and computes a similarity score between them. The model is composed of a convolutional network to encode the sentences into a sentence embedding, and a projection matrix which projects the visual feature into the same space as the sentence embedding. The cosine similarity measure is used to evaluate the similarity between the sentence embedding vector and the projected visual feature vector.

The convolutional network we use to encode sentences is based on the paper [32], where it is used for sentence sentiment prediction. Figure 5.2 shows the block diagram of our CNN-based evaluator. Here, the input sentence is represented as a sequence of word vectors, which can either be statically initialized with some standard word vector encodings, such as GloVe [45] or word2vec [42], or learned during the training phase. These word vectors are fed into a convolutional neural network which computes an encoding of the sentence.

The first layer in the CNN consists of convolutional filters of different sizes. All the filters operate over the entire word vectors, but vary in the number of words they cover, i.e., each filter is of the size $n \times W_{\text{dim}}$. Here, n is the n-gram over which filter operates and W_{dim} is the dimensions of the word vector representation used in the CNN. For example, one can have filters operating over bigrams, trigrams, etc. Specifically, the CNN evaluator

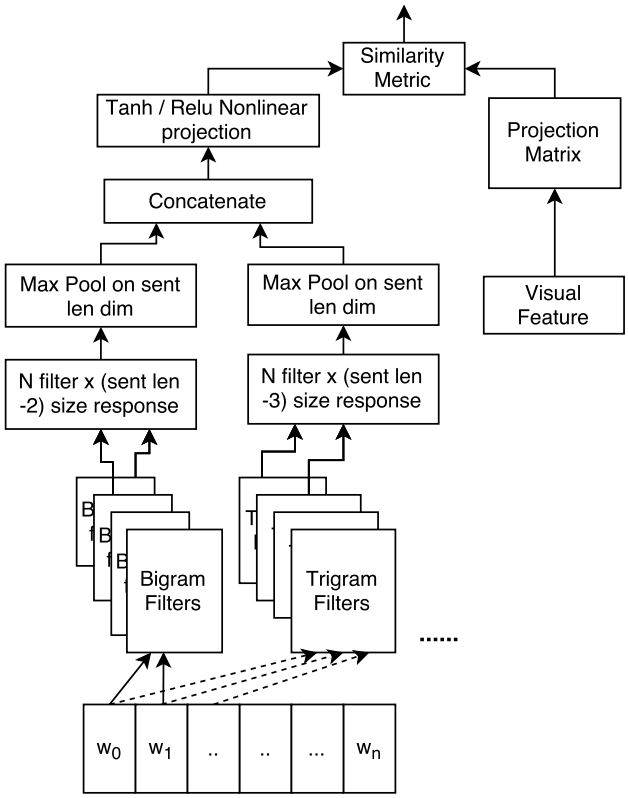


Figure 5.2: CNN based evaluator network to compute the similarity between an image and a caption.

used in this work has filters with bigrams, trigrams, 4-grams and 5-grams. Additionally, F_n number of filters of each n-gram type is used.

The filter outputs are max-pooled, which reduces the filter response over entire sentence into a single scalar, i.e., its maximum response. We can therefore think of each filter as looking for a specific n-gram, disregarding its location within the sentence. These pooled outputs are concatenated and then projected to the desired vector size to produce the final sentence encoding.

5.4.2.1 Training the Evaluator Network

The evaluator network needs to be trained to assign a high score for the best caption and lower scores for other captions. For each training set image or video V, the CNN evaluator scores the corresponding ground truth caption, S^+, and k negative samples, $S_i^-, i = 1, \ldots, k$, drawn randomly from the ground truth captions of other samples in the training dataset. Now the

training cost function C is devised to maximize the score for the positive sample and to minimize it for the negative samples. This is achieved by applying a softmax on the scores of this batch (one positive and k negative samples) and maximizing the softmax score of the positive sample:

$$P(S^+|S^-,V) = \frac{\exp(-f(S^+,V))}{\exp(-f(S^+,V)) + \sum_i^k \exp(-f(S_i^-,V))} \tag{5.10}$$

$$C = -\log P(S^+|S^-,V)\ . \tag{5.11}$$

Equation (5.10) shows this computation with $f(S,V)$ representing the similarity metric between the sentence candidate S and media V. In our current method, we use the cosine similarity between the two vectors for the purpose of $f(S,V)$.

5.4.2.2 Utilizing the Evaluator Network

Once trained, the evaluator network can be utilized to compute the similarity for a visual-input–caption pair. In this work, it is used to ensemble a set of language models. Concretely, every caption, S_i, in the candidate set, C_p, is paired with the visual input and is fed to the evaluator network. This gives us p similarity scores $(E_{cnn}(S_0,V),\cdots,E_{cnn}(S_p,V))$, one for each candidate caption, S_i. The candidate with the highest similarity is chosen as the output caption from the ensemble for the visual input.

$$S_{cnn} = \arg\max_{S_i} E_{cnn}(S_i,V) \tag{5.12}$$

Chapter 6

Experiments and Results

So far, we have discussed several feature extraction methods and alternate language model architectures for automatic caption generation. Now, it is time to put these proposals to test and evaluate how well they perform on the captioning task. In this chapter the results from the experiments on three separate datasets, one for image captioning (MS-COCO) and two for video captioning (LSMDC and MSR-VTT), will be discussed. We will rely on the four evaluation metrics described in Section 3.2, namely BLEU, METEOR, ROUGE-L and CIDEr, to evaluate the performance of our models on the captioning task. Additionally, on validation sets, we also compute the models' perplexities of the ground truth captions and use this as a fifth metric of performance.

As already discussed, these automatic evaluation metrics only approximately track the human judgments of how good the generated captions are. To meticulously evaluate the captioning models, we need to collect human judgments on the captions generated by them. However, it is a costly exercise to collect human evaluations for every model that needs to be compared. Luckily enough, human judgments were collected and used to evaluate such systems in the captioning challenges held over the course of the last year. Few of the best video captioning models presented in this thesis were also submitted to two such video captioning challenges. The human evaluations obtained from the two challenges will be presented here, and the results of the competitions will be summarized.

Since the COCO dataset is the largest of the three datasets we have used, we conduct all the language model experiments on this dataset. These experiments include the performance analysis of our proposed language model extensions namely, addition of persist features, increasing depth with residual connections and utilizing hierarchical decoder is conducted on the COCO dataset. Also, since the language model used in video captioning has the

same architecture as the one used in image captioning, it is assumed that the same results will hold there too. Thus we limit our experiments in the video captioning task to the evaluation of different video features.

Finally, a note on the notation used to represent different models presented in this chapter. We will use the notation "δ#" to denote different models with δ representing the dataset on which the model is trained and # being the model number within the dataset. Thus, δ ='C' for the models trained on the COCO dataset, δ ='L' for models trained on the LSMDC dataset and δ ='M' for ones trained on the MSR-VTT dataset. In all the tables, the model name appears under the columns named '#'.

6.1 Implementation Details

Before getting into the evaluation, we will first briefly discuss some implementation platform details, and some hyper-parameter choices.

Language Model: The proposed LSTM language model used for both image and video captioning is implemented using the Theano library [4]. Theano allows to run the computations on a GPU with minimum effort, leading to much quicker training and testing cycles. The language models are trained using stochastic gradient descent with the RMSProp [22] algorithm and the dropout regularization is implemented as described in [78]. The error is back-propagated to all the language model parameters and word embedding matrices, but the image feature extraction models are kept constant.

The language model is trained by minimizing the negative log-likelihood assigned by the model to the training samples. But, as noted before, the evaluation of the trained model is done using the automatic evaluation metrics. This discrepancy between the training objective and the evaluation cost is forced upon us as the automatic metrics themselves are not differentiable and it would thus be hard to train a model directly to optimize these metrics. Still, we can justify using the perplexity as the training cost, by empirically observing that the perplexity measure roughly tracks the automatic metrics. Concretely, optimizing the model parameters to minimize perplexity also improves the model performance as per automatic metrics. To verify this, a quick experiment was conducted where the perplexity measure is computed and recorded on the validation and training sets using the intermediate models after every epoch of training. Simultaneously, these intermediate models are used to generate captions on the validation set and automatic metrics are computed on these generated captions. Results are plotted in Figure 6.1.

We see that perplexity indeed tracks the performance on the other evaluation metrics very well.

In all our experiments we use beam search to generate sentences from a trained model. After experimenting with different beam sizes, we found that the beam size of $b = 5$ works well across all our models on the COCO and the MSR-VTT datasets, whereas $b = 1$ worked better on the LSMDC dataset.

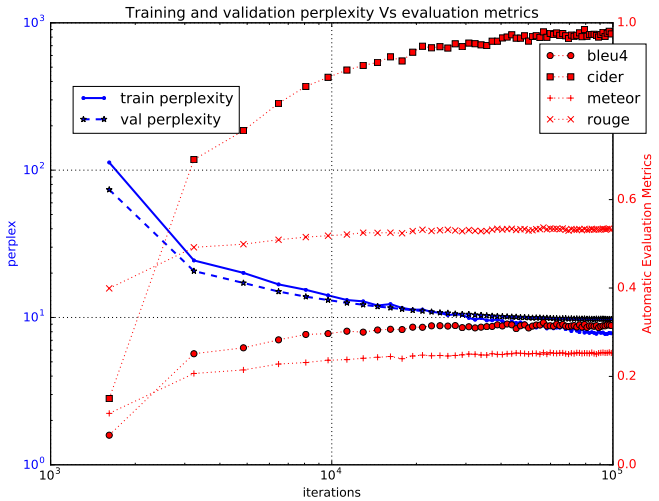


Figure 6.1: Evolution of training and validation perplexity and automatic metrics computed on the COCO validation set during training.

Visual Feature Extraction: The CNN feature extraction and the Faster R-CNN models are based on the Caffe library [26]. Using the code released for the Faster R-CNN network[1], we train it on the MS-COCO dataset to detect the 80 object categories annotated in COCO. Then, this detector is run on the entire COCO dataset and the spatial map feature vectors are created using the bounding boxes output by the detectors.

Dense trajectory feature extraction on videos was done using the source code provided by the authors of [66]. To extract the C3D features, the code and the model pre-trained on the Sports-1M dataset provided by the authors of [61] was used.

CNN Evaluator: The proposed CNN evaluator is also implemented using the Theano library. It is created with bi-, tri-, 4-, and 5-gram filters and $F_n = 100$ filters of each type. The word vectors used in the evaluator are

[1] `https://github.com/rbgirshick/py-faster-rcnn`

chosen to have $W_{dim} = 100$ dimensions. In order to train the CNN evaluator, each ground truth caption–image/video pair needs k negative captions as well. In order to optimize the training process, we sample $k + 1$ random image caption pairs and treat the other k captions as negative samples for each image. This is much faster than sampling separate negative captions for each image-caption pair, and improved the overall training speed. In our experiments, we found setting $k = 49$ achieved good results.

6.2 Image Captioning

In this section we report results of our experiments on image captioning conducted on the MS-COCO dataset. In the first subsection we will discuss the MS-COCO dataset in detail. Next the results from the local evaluation conducted on the COCO validation set, comparing various choices for language model architectures and image features, are reported. This is followed by results from the test set obtained from the Microsoft CodaLab portal and a comparison to some best published works on this dataset.

6.2.1 MS-COCO Dataset

In all our image captioning experiments we use the Microsoft COCO 2014 dataset [39] for training and evaluation. This dataset consists of 164,062 images split into training set of 82,783 images, validation set of 40,504 images and test set of 40,775 images. An additional 40k test images were released in the 2015 version of the dataset, but it is not used in the experiments reported in this thesis. The training and validation sets have five reference captions for each image annotated by humans. The total 413,915 reference captions from the training set have 23,528 unique words.

The COCO dataset also has object segmentations available for each image and for objects belonging to 80 specified categories. The categories consist of common object types such as *person*, *car*, *bus*, *skateboard*, etc. We make use of the object segmentations when training the Faster Region-based Convolutional Neural Networks for extracting the object location features, as described in Section 4.1.4. The segmentation information is, however, not utilized when training the LSTM language model nor when using it with the validation and test images.

Before using the reference captions in the COCO dataset for training, we tokenize the text and remove symbols and numerals. Words occurring less than five times are also removed in order to weed out spelling mistakes

and extremely rare words, for which we have insufficient data to learn. This leaves us with a training corpus vocabulary consisting of 8,790 unique words.

To evaluate the utility of the proposed set of image features and LSTM network architectures, we use the COCO 2014 validation set and the five reference sentences available for all images in it. Microsoft COCO team has also made an evaluation server available on CodaLab[2] where researchers can upload their captions for the test set and view the resulting evaluation metrics. We use this portal to evaluate our models on the COCO Image Captioning Challenge 2014 test set. Here, we also compare our performance against other well-performing or state-of-the-art entries in the CodaLab leaderboard.

6.2.2 Results on Validation Set

In Chapters 4 and 5 we discussed several image features and language model extensions, respectively, to improve the captioning system over the baseline model. In order to obtain the best possible captioning system for the COCO dataset, we need to measure the performance of these different features and language model combinations. Since training a model for every combination of a feature and a choice of language model parameters is prohibitively expensive, separate experiments are conducted to determine the best features and the best language model choices, while keeping the other fixed. We observe that these two aspects are fairly independent and choosing the best feature and the best language model independently gives us the best captioning model.

6.2.2.1 Evaluating the Init and Persist Paths

#	Features		Performance metrics				
	init	persist	BLEU-4	METEOR	ROUGE-L	CIDEr	Pplx
C1	gCNN	—	0.289	0.238	0.514	0.860	10.43
C2	gCNN	SVM80	0.292	0.239	0.516	0.871	10.34
C3	gCNN	gCNN	**0.302**	0.243	**0.523**	0.897	10.25
C4	SVM80	gCNN	**0.302**	**0.244**	**0.523**	**0.909**	10.30
C5	SVM80	SVM80	0.261	0.225	0.492	0.785	10.78

Table 6.1: Evaluating the utility of the init and persist input channels to the LSTM language model on COCO validation set.

In Chapter 5 we introduced a new input to the LSTM language model, the *persist* path, positing that it is beneficial for the language model to

[2] `https://competitions.codalab.org/competitions/3221`

have access to the visual features throughout the caption generation process. This new input also enables us to provide two different features as input to the language model. Table 6.1 presents the results from experiments trying to determine the best use for the two input channels, *init* and *persist*. The columns *init* and *persist* indicate what visual features were used as the initializing and persistent inputs to the language model respectively. In these experiments, our language model has only a single layer of LSTM cells with both the word-embeddings and the LSTM layer being of 512 dimensions.

Our baseline model, C1, only uses the *gCNN* (from GoogLeNet) image features as the *init* input. Compared to this, additionally providing the 80 dimensional detector features, *SVM80*, using the *persist* channel improves the performance slightly, in model C2. Instead, if we provide the *gCNN* features to both the inputs, as in model C3, there is a dramatic improvement in the performance in all the four metrics, as well as in the validation perplexity. This tells us that it is beneficial for the language model to have access to the CNN features throughout the caption generation process.

Instead of redundantly using the same *gCNN* features in both *init* and *persist* paths, we can now replace the *init* feature with the *SVM80* feature to get a marginal performance improvement as seen in model C4. Model C5 tells us that using only the *SVM80* features is not good and the CNN features need to be presented in the *persist* to get the best performance.

6.2.2.2 Finding the Best Image Features

#	Init feature	Performance metrics				
		BLEU-4	METEOR	ROUGE-L	CIDEr	Pplx
C4	SVM80	0.302	0.244	0.523	0.909	10.30
C6	FRC80	0.316	0.249	**0.534**	0.952	10.15
C7	SUN397	0.301	0.241	0.521	0.894	10.40
C8	SUN397⊕FRC80	0.315	**0.250**	0.532	0.954	10.05
C9	4×4IoU	0.302	0.244	0.522	0.913	10.21
C10	4×4Gauss	0.308	0.246	0.527	0.921	10.15
C11	3+3Gauss	0.308	0.247	0.527	0.928	10.08
C12	3+3Gauss⊕SUN397 ⊕FRC80	**0.318**	**0.250**	0.533	**0.957**	**9.93**

Table 6.2: Evaluating the efficacy various image features using fixed language model configuration on COCO validation set.

Next we report from the experiments to determine the best image features to pair with the CNN features. Using the results from previous subsection as

guideline, we keep the gCNN features fixed as *persist* input in the following experiments, and only change the feature input in the *init* channel. The language model parameters also remain the same as Section 6.2.2.1. Table 6.2 presents the results from these experiments. Note that, here we use the "$\oplus$" symbol to denote the vector concatenation operation.

Comparing the results of models C4 and C6, we see that the *FRC80* features outperforms the SVM80 features with a specially significant gain in the CIDEr metric. The Faster R-CNN based object features thus seem to overcome the simpler SVM detector output based features. This also supports the hypothesis that good explicit object detectors can effectively complement the CNN image features. The object detectors are trained to detect multiple objects explicitly, and although they don't encode any information about the object shape or other attributes, just the information about the probability of occurrence of different objects seems very beneficial to the captioning task.

Using the scene detection features, *SUN397*, alone as *init* input in model C7 worsens the performance. However, augmenting the *FRC80* object features with scene information by concatenating *SUN397* features, as shown in model C8, improves the performance over C6 in 3 metrics.

Next we compare the spatial grid features using models C9, C10 and C11. We find that using the integral of Gaussian (4.2) performs better than using the intersection-over-union (IoU) measure (4.1) when constructing these features as seen by comparing C9 and C10. In general, however, the spatial grid features do not match the performance of the *FRC80* features, even though *FRC80* only encodes a subset of the information represented in the spatial grid features. This could be due to the fact that the spatial grid features are of much higher dimension than the *FRC80* feature vectors. This hypothesis is also strengthened by observing that model C11, which uses smaller *3+3Gauss* features, performs the best among the models using the spatial grid features.

Next we train model C12 by concatenating the *FRC80*, *SUN397* and *3+3Gauss* features. This model now has access to object detection, scene type and object location information apart from the CNN features, and is our best-performing model with this language model configuration.

6.2.2.3 How Deep Should We Go?

Table 6.3 presents the results from experiments with the depth of the LSTM language model. Column *depth* specifies the number N of LSTM layers in the model, with N-res being an LSTM network with N layers and residual connections. For these experiments, the LSTM layer size and word encoding size are still kept at 512 dimensions, but only the number of LSTM layers is

#	Depth	Performance metrics				
		BLEU-4	METEOR	ROUGE-L	CIDEr	Pplx
C8	1	0.315	0.250	0.532	0.954	10.05
C13	2	0.318	0.252	0.535	**0.967**	10.14
C14	3	0.316	0.253	0.533	0.964	10.34
C15	4	0.316	0.250	0.533	0.956	10.69
C16	2-res	**0.320**	0.253	**0.536**	0.966	9.92
C17	3-res	0.316	**0.254**	0.532	0.962	**9.69**
C18	4-res	0.316	0.253	0.535	0.964	9.75

Table 6.3: Results from experiments with language model depth, with fixed input features on COCO validation set.

varied. The *SUN397⊕FRC80* features are used as the *init* input and *gCNN* features are used as the *persist* input.

When we increase the number of layers without adding residual connections, we see that the perplexity metric worsens, although there is moderate improvement in the other metrics up to the depth of three layers. This is seen comparing model C8 with models C13–C15. However, when we increase the depth to four layers, we see that it performs similar to the single layer model in automatic evaluation metrics with perplexity being significantly worse.

Adding the residual connections significantly improves the perplexity of the validation set while the performance on the metrics improves slightly. The biggest gain is seen in the CIDEr metric. We also find that the residual connections improve the training convergence speed. This is illustrated in Figure 6.2, where we show the progression of the perplexity measure during the training of the 4-layered models with (C18) and without (C15) residual connections. We see that the performance gain seems to saturate by four layers even with residual connections. So the choice for the best architecture is between the 2- or 3-layered model with residual connections, considering all the five metrics. But since the 3-layered model, C17, produces more diverse captions, as seen in the Section 6.2.3 this configuration is chosen for further experiments.

6.2.2.4 Hierarchical Decoder

Previously, we determined the 3-layered model with residual connections, C17, to be the best language model configuration. We further upgrade this model by changing the *init* feature to *3+3Gauss⊕SUN397⊕FRC80* in model C19 shown in Table 6.4. This slightly improves the performance, with model C19 improving over model C17 in three measures. We use this configuration

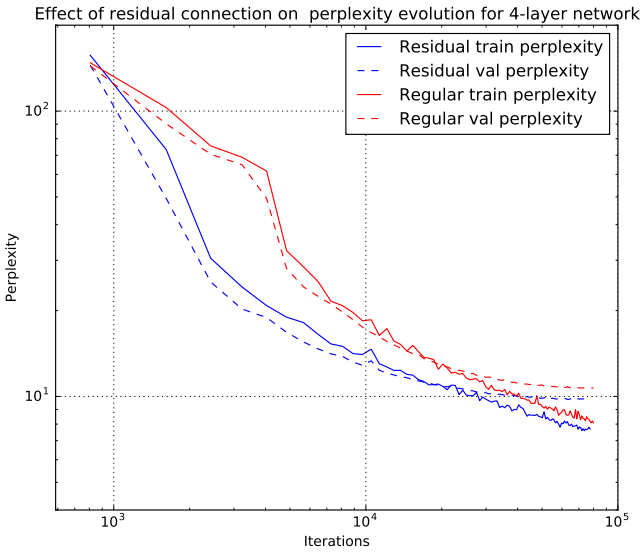


Figure 6.2: Comparing the evolution of perplexity during training of a 4-layered model with and without residual connections on COCO dataset.

#	Class clustering	Performance metrics				
		BLEU-4	METEOR	ROUGE-L	CIDEr	Pplx
C17	–	0.316	**0.254**	0.532	0.962	**9.69**
C19	–	**0.319**	0.252	**0.535**	**0.970**	9.72
C20	K-Means, 200 Class	0.286	0.245	0.523	0.906	10.10
C21	Brown, 200 Class	0.286	0.245	0.523	0.906	10.10

Table 6.4: Results from models using a hierarchical decoder.

as the basis to run our experiments with the hierarchical decoder presented in Section 5.3. These results are also presented in Table 6.4. Both the models with hierarchical decoder shown in Table 6.4, C20 and C21, use the same *init* and *persist* features as C19 and have three layers connected with residual connections.

Model C20 is trained with the hierarchical decoder in which the words are clustered to classes using the K-means method. For this, we run K-means on word vectors learned by model C19, with K=200 to obtain 200 classes. Alternatively, C21 uses class assignments obtained from the Brown clustering algorithm run on the training corpus, again with 200 classes.

We see that both these models perform similarly, with C21 only slightly better than C20 in CIDEr metric and perplexity. More interestingly, both the models are considerably worse than the corresponding simple decoder

model, C19, in all the performance metrics. On manual inspection of the captions produced by them, we find that both C20 and C21 make significantly more grammatical errors, with often missing conjunctions and stop words. But on the bright side, the models using the hierarchical decoder generate significantly more diverse and descriptive captions as will be seen in Section 6.2.3. Thus, it seems that the hierarchical decoder indeed makes the captions richer, but it also adversely affects the correctness of the captions generated.

6.2.2.5 Ensembling

#	Method	Evaluators	Eval type	Performance metrics			
				BLEU-4	METEOR	ROUGE-L	CIDEr
C19	–	–	–	0.319	0.252	0.535	0.970
C22	Self-Eval	self	max	0.319	0.253	0.533	0.966
C23	Mutual-Eval	All six	max-mean	0.303	0.249	0.526	0.925
C24	Mutual-Eval	three best	max-mean	0.304	0.250	0.527	0.932
C25	Mutual-Eval	all six	max-max	0.305	0.251	0.528	0.935
C26	Mutual-Eval	three best	max-max	0.305	0.251	0.528	0.939
C27	CNN Evaluator	CNN	max	**0.320**	**0.254**	**0.536**	**0.978**

Table 6.5: Comparison of different ensembling techniques.

Next results from the experiments ensembling multiple caption generating models will be discussed. As a first step, six models are chosen to form the model set participating in the ensemble. The six models used here include C6, C14, C17, C19, and two models trained using the *SUN397* concatenated with one of the two spatial grid features, 3+3Gauss and 4×4Gauss, respectively. The models in the ensemble were chosen to include the best-performing models while also maintaining diversity of architectures in the ensemble.

Model C22 picks the best candidate among the pool of candidate captions using the *Self-Eval* method discussed in Section 5.4.1. Models C23 to C26 are ensembles based on the *Mutual-Eval* method. Here, the models mentioned in the *Evaluators* column assign probability to all the candidate captions. In case of *all six* all the models participating in the ensemble are used to rate all the candidates. In contrast, in *three best* three models with the best perplexity scores were used to rate the candidates. Once candidate captions are scored, the technique specified in the *Eval type* column is used to aggregate these probability scores and pick the best candidate. Finally, C27 is the ensemble model which uses the CNN-based evaluator to rate and pick the candidates.

Comparing their performance based on the evaluation metrics, we see that only the CNN evaluator based C27 model improves over the best single model participating in the ensemble, C19. Other two ensembling techniques fall short, with the self-evaluation based method doing better than any of the mutual-evaluation based techniques. Among the mutual evaluation based techniques, we see that using only the three best models is better than using all the six models as evaluators. We also see that "max-max" works slightly better than "max-mean" as a method to aggregate the scores assigned to a caption. However, the mutual evaluation based and CNN-based ensembles produce much more diverse captions than both the single models and the self-evaluation based ensembles, as we will see in Section 6.2.3.

6.2.3 Language Diversity Analysis

#	Mean Length	Vocabulary Size	% Unique Captions	% New Captions	Comments
C1	9.27	814	16.10	11.76	*init* vs
C4	9.08	923	22.42	17.23	*persist*
C8	9.02	962	23.23	18.25	
C16	9.11	983	26.39	20.80	varying
C17	9.18	1197	31.14	24.03	depth
C18	9.23	1164	31.10	24.28	
C19	9.01	1112	28.43	22.04	regular vs
C21	**9.58**	1191	**49.16**	**44.39**	factorized
C22	9.06	993	21.34	15.36	
C26	9.38	**1380**	41.65	33.64	ensemble models
C27	9.13	1303	40.35	32.33	

Table 6.6: Language diversity statistics of a few selected models.

The evaluation based on the automatic metrics presented in the previous sub-section despite being the accepted standard in the literature, is not very intuitive and does not give an idea of how rich the captions generated by the models are. To address this, we will look at few simple metrics which measure the diversity of the captions generated by the models. Specifically, we generate captions to the 40,504 images in the COCO validation set by using different models and look at four different heuristic measures of diversity in these captions. These heuristic measures are 1) the mean length of the generated captions, 2) the size of the vocabulary used by the models, 3) the percentage of unique captions generated, and 4) the percentage of generated captions not seen in the training set.

Table 6.6 shows these statistics computed for various models. Comparing the statistics for C1 and C4 we see that adding the *persist* path moderately improves vocabulary size and diversity of the captions generated, but with the average length of the captions dropping slightly. Next, comparing models C8, C16, C17 and C18, which vary only in the depth of the language model, we see that increasing depth improves all the four statistics, with the 3-layered model generating $\sim 24\%$ new captions.

The biggest gain in sentence diversity is seen when we use the hierarchical decoder based model, C21. Comparing it to equivalent model C19, we see that C21 generates almost twice as many new captions, using a vocabulary of approximately the same size.

Ensembling of multiple language models also helps improve the vocabulary size and diversity of the captions, with the exception of model C22. Model C22, based on self-evaluation, seems to prefer unoriginal captions and common words, with the percentage of new sentences dropping to just $\sim 15\%$. This is the second lowest of all the models presented in Table 6.6. Contrastingly, both model C26, based on mutual evaluation, and model C27, based on CNN evaluator, pick almost double the number of new captions. They also have significantly larger vocabulary and rival the C21 model in terms of caption diversity.

Considering both the language diversity statistics in Table 6.6 and automatic evaluation metrics in Table 6.3, we conclude that model C27 is our best captioning model on the MS-COCO validation set.

6.2.4 Comparison With State-of-the-Art

Leaderboard Name	**BLEU-4**		**METEOR**		**ROUGE-L**		**CIDEr**	
	c5	c40	c5	c40	c5	c40	c5	c40
AugmentCNNwithDet (C19)	0.315	0.597	0.251	0.340	0.531	**0.683**	**0.956**	**0.968**
—— (C27)	0.310	0.596	0.250	0.338	0.529	0.681	0.948	0.961
ATT_VC [75]	**0.316**	0.599	0.250	0.335	**0.535**	0.682	0.943	0.958
—— (C17)	0.309	0.588	0.251	0.342	0.529	0.680	0.943	0.948
OriolVinyals [65]	0.309	0.587	**0.254**	**0.346**	0.530	0.682	0.943	0.946
MSR_Captivator [16]	0.308	**0.601**	0.248	0.339	0.526	0.680	0.931	0.937
Berkeley LRCN [14]	0.306	**0.585**	0.247	0.335	0.528	0.678	0.921	0.934
human [9]	0.217	0.471	0.252	0.335	0.484	0.626	0.854	0.910
Montreal/Toronto [72]	0.277	0.537	0.241	0.322	0.516	0.654	0.865	0.893

Table 6.7: COCO 2014 test results. The scores are based on Microsoft COCO leaderboard. c5 *and* c40 *indicate the number of reference captions used in evaluation. The models are sorted based on CIDEr score as in the leaderboard.*

We compare the performance of our models with several state-of-the-art models reported on the COCO 2014 captioning leaderboard and published results on the validation set. For this purpose, we submitted captions form models C17, C19 and the CNN ensemble model C27 to the CodaLab portal.

We compare our results with ATT_VC [75], MSR_Captivator [16], Berkeley LRCN [14], Montreal/Toronto [72], OriolVinyals [65], and human [9] reported in CodaLab. It is worth noting that the OriolVinyals model shares the same architecture with our adopted baseline model C1.

Table 6.7 reports the results of the benchmark[3]. We compare our model with the other submissions within the top ten ranks in the COCO leaderboard, as per METEOR metric, excluding the entries not associated with any published work or report. Note that each metric in Table 6.7 has two columns, *c5* and *c40*. This is because COCO dataset contains 40 reference captions for a small subset of images in the test set, referred to as *c40*. As already mentioned, using a larger number of reference captions makes the metrics better correlated with human judgments and thus the metrics obtained on the *c40* are more reliable. The *c5* metrics are obtained from the regular test set with only five reference captions.

We can see that the performances of our models drop slightly on the test set compared to that on the validation set. Also, while the ensemble model was the best model on the validation set, we observe that C19 is a better model on the test set. This could be because some models in the ensemble do not generalize well to the test set. Considering the overall performance of our model C19, we are outperforming several state-of-the-art published results. More recently, there are three new entries on the leaderboard which are doing better than our C19 model, but since they are not associated with any published work, we refrain from discussing them here.

We should note here that the scores in CodaLab leaderboard do not necessarily reflect the original published work results due to changes and updates. Thus, we also present a comparison with the published scores on the validation set in Table 6.8. We see that our models outperform all published results on the validation set, with the largest improvement seen in the CIDEr score. We could attribute most of this improvement to the rich set of image features we use compared to other methods, most of which rely solely on CNN image features.

#	BLEU-4	METEOR	ROUGE-L	CIDEr
C27	**0.320**	**0.254**	**0.536**	**0.978**
C20	0.319	0.252	0.535	0.970
ATT_VC [75]	0.304	0.243	–	–
Berkeley LRCN [14]	0.300	0.242	0.524	0.896
OriolVinyals [65]	0.277	0.233	–	0.855
MSR_Captivator [16]	0.257	0.236	–	–
Montreal/Toronto [72]	0.250	0.230	–	–

Table 6.8: Comparison of our models to the best published results on COCO 2014 validation set.

6.2.5 Qualitative Examples and Discussion

So far, we evaluated our models using the automatic metrics, but the metrics are only approximative in judging the correctness of the captions. Figure 6.3 shows some sample captions generated by the ensemble model C27 and the best single model C19 on images from the validation set. We can see that the generated captions are often fairly accurate, but still sometimes contain quite rudimentary errors. For the top left image, the CNN evaluator manages pick out a caption which captures all the three objects in the image. In comparison, C19 only mentions "man" and "sheep". Similarly, in all the images in the top row, the CNN evaluator picks the caption with more details than the one produced by C19. In contrast, the bottom row contains examples where the caption generated by C19 describes the image better than the one picked by C27.

One of the errors which still persists is that our models are not able to precisely learn the relationships between the objects in an image. For example, our models find it hard to differentiate between a person riding a bicycle versus just standing next to it, as seen in bottom right image in Figure 6.3. A similar pattern of error is seen in relationships between various object types.

Another commonly occurring error happens in counting, wherein the captions tend to get the number of objects wrong. We also see that some generated captions repeat the words referring to objects instead of using numerals. Still one drawback is in the vocabulary size of the generated sentences. As already seen in Table 6.6, even the ensemble model uses only 1,303 words which is about $1/8^{th}$ of the words presented to the models during the training.

[3]The leaderboard with more models and scores is at `http://mscoco.org/dataset/#captions-leaderboard`

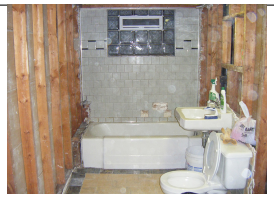
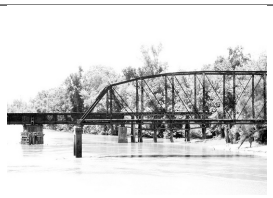
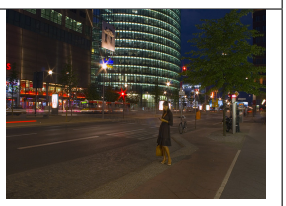
			
C27:	a man and a dog herding sheep in a field	a bathroom with a sink toilet and bathtub	a bottle of wine and a glass of wine
C19:	a man standing next to a herd of sheep	a bathroom with a toilet and a sink	two bottles of wine sitting on a table
			
C27:	a view of a bridge in the snow	a table with plates of food on it	a person riding a bike down a city street
C19:	a train crossing a bridge over a river	a table topped with plates of food and drinks	a city street filled with lots of traffic

Figure 6.3: Captions generated for some images from the COCO validation set by two of our models. The first row contains samples where the ensemble model, C27, performs better, and the second row cases where C19 is better.

6.3 Video Captioning

Next we will shift our attention to evaluating our video captioning models. As discussed before, our video captioning datasets are much smaller in terms of training video–caption pairs compared to the COCO dataset. Thus we will focus our efforts here on identifying the best video features suitable for the video captioning task, with majority of our experiments conducted on the richer MSR-VTT dataset. While choosing the language model configuration for these experiments, we will rely heavily on the insights obtained from the experiments on the COCO dataset.

We extract the *gCNN* and *SVM80* features only on the keyframe in the LSMDC dataset and on one frame every second in the MSR-VTT dataset. The features extracted only from the keyframe will be indicated with the prefix *kf-* and features extracted from frames every second will be denoted by the prefix *ps-*. On the LSMDC dataset, we only use the dense trajectory features (DT) as segment-level features. On MSR-VTT dataset, apart from DT features we also experiment with the improved dense trajectory features (IDT), and the features extracted from the C3D network.

The evaluation of the video captioning models also rely on the same four

evaluation metrics and model perplexity measure on validation set as used for image captioning. On the test sets, both the LSMDC and the MSR-VTT challenge organizers provide evaluation using these metrics and the human judgments, which we will present here.

6.3.1 Datasets

First, we will discuss the LSMDC and MSR-VTT video captioning datasets in detail here.

6.3.1.1 LSMDC Dataset

The LSMDC dataset is the combination of two movie description datasets, MPII-MD [50] and M-VAD [60]. Both the MPII-MD and M-VAD datasets contain clips extracted from movies and a single reference caption describing the clip, but they differ in the source of the reference caption. MPII-MD was collected from blue-ray movie clips and corresponding audio description (AD) and script data. The AD data was transcribed to text and manually aligned to the video clips. M-VAD was collected from DVD movie clips and only relies on AD data to annotate the clips with a reference caption. In both datasets, any mention of people names in the collected reference captions has been replaced with the token "SOMEONE".

Combining these two, LSMDC dataset has 118,081 video clips extracted from 202 unique movies and one reference caption for each clip. These clips are split into training set (91,908), validation set (6,542), public test set (10,053) and a blind test set (9,578). The average length of these clips is just 4.8 seconds and the captions in training set contain 22,829 unique words. This is filtered down to a vocabulary of 8,814 words, again by removing words occurring less than five times.

We have to note here that the reference captions on the LSMDC dataset are relatively noisier than the other two dataset we have used. They have problems due to misalignment of the captions with the video clips, where the caption could be referring to objects which occur just before or after the point the clip was cut from the movie. Also, in some cases indiscriminate replacing of names with "SOMEONE" can lead to very ambiguous captions. We can also see the effect of this in the poor performance of our models and other state-of-the-art models on this dataset.

6.3.1.2 MSR-VTT Dataset

The MSR-VTT dataset [71] consists of 10,000 video clips with 20 human-annotated captions for each of them. Each video belongs to one of 20 categories including *music*, *gaming*, *sports*, *news*, etc. The dataset is split into training (6,513 clips), validation (497 clips) and test (2,990 clips) sets. The training set videos are between 10 to 30 seconds long, with most of them less than 20 seconds in duration.

The reference captions come from a vocabulary of ∼30k words, which is reduced to a vocabulary of ∼8k after removing words occurring fewer than five times. Although the number of video clips in the dataset is relatively small compared to M-VAD [50] and MPII-MD [60], the dataset is attractive due to the diversity of its videos and due to the larger number of reference captions per video.

By visually examining the training set, one can find many different video styles, including slide shows with static images, recordings of computer monitor showing users playing video games, movie trailers with lots of fast cuts, smooth single shot videos involving activities such as cooking, news, interviews, etc. This means that the video captioning algorithm applied to this dataset needs to handle all these diverse video styles. If one relies only on action recognition based features, the approach will suffer with videos with lots of cuts and scene changes. On the other hand, if one relies only on frame-based features, the system will fail to identify fine-grained differences in diverse action-oriented categories such as sports or cooking. Consequently, in this thesis an ensemble based approach is taken to build a video captioning system on this dataset. The solution consists of multiple captioning models, each trained on different types of features, and a CNN-based evaluator network to pick the final candidate caption.

6.3.2 LSMDC

The keyframe used to extract *kf-gCNN* and *kf-SVM80* features is sampled from the center of the video after the padding video clips to be of at least two seconds long. In case of LSMDC dataset, it is reasonable to assume that the single keyframe is quite representative of the video clips, as the clips are very short.

6.3.2.1 Results on Public Test Set

To evaluate various forms of the video captioning models presented here, the LSMDC 2015 public test set is used as the benchmark. Table 6.9 shows the

#	init	persist	perplex	BLEU_4	METEOR	ROUGE_L	CIDEr
C4	kf-SVM80	kf-gCNN	–	0.003	0.053	0.114	0.052
L1	kf-gCNN	–	56.08	0.004	0.058	0.140	0.071
L2	kf-gCNN	kf-SVM80	60.78	0.004	**0.060**	0.142	0.073
L3	kf-SVM80	kf-gCNN	59.07	0.005	0.059	0.144	0.087
L4	DT	–	54.89	0.005	0.057	0.145	0.087
L5	DT	kf-SVM80	59.75	0.005	0.057	0.141	0.081
L6	kf-SVM80	DT	55.14	**0.006**	0.058	**0.146**	**0.092**
L6*	kf-SVM80	DT	55.14	0.004	0.049	0.128	0.082

Table 6.9: Results obtained on the public test set of LSMDC 2015. The model L6 is identical to L6 except that it uses beam size $b = 5$. All the other models use beam size $b = 1$*

four evaluation metrics computed for different models.

In order to get a quick baseline, the C4 model trained on the COCO dataset is used to generate captions on the LSMDC test set. This model was chosen as it is the best model using the *gCNN* and *SVM80* features, which are compatible with keyframe-wise *kf-gCNN* and *kf-SVM80* features. The captions generated from the C4 model are translated with a simple rule-based translation to better match the LSMDC vocabulary. The translation is implemented using the simple $w_{\text{in}} \longrightarrow w_{\text{out}}$ rule:

$$w_{\text{out}} = \begin{cases} \text{SOMEONE}, & \text{if } w_{\text{in}} \in \{\text{man}, \text{woman}, \\ & \qquad \text{person}, \text{boy}, \text{girl}\} \\ w_{\text{in}}, & \text{otherwise.} \end{cases} \tag{6.1}$$

As we can see in Table 6.9, the performance of the COCO model on the LSMDC dataset is quite bad. This is caused by two factors, first the vocabulary used in the LSMDC captions is quite different from that of COCO and hence the automatic evaluation will rate the COCO vocabulary-based captions quite badly. Another factor is that motion related information is completely ignored in this model.

Next, new models are trained using the reference captions in the LSMDC dataset and keyframe features. Comparing models L1, L2 and L3 in Table 6.9, we see that the configuration with *SVM80* features as *init* input and *gCNN* features as *persist* input performs the best. This matches with the similar observation on COCO dataset. We can see that these models clearly outperform the COCO baseline, mainly due to the vocabulary update.

Finally, three more models are trained using the dense trajectory features and the keyframe-based *SVM80* features, presented in Table 6.9 as models L4–L6. Again we see that using the higher-dimensional feature, here the

DT feature, as the *persistent* input to the LSTM network gives the best performance among this group of models. Comparing model L6 with L3 shows that using video features as opposed to just keyframe features gives a better performance. We can see that model L6 benefits from combining both keyframe and trajectory features as opposed to just using the trajectory features as in model L4. The result of model L6 can be regarded as the best one obtained in our LSMDC experiments: it has the best scores in three out of four metrics. Therefore, the captions generated by the model L6 on the blind test set were submitted to the LSMDC 2015 Challenge.

A rather surprising finding from the experiments on the LSMDC dataset is that, using larger beam sizes in inference lead to poorer performance. This is seen from comparing models L6 and L6* in Table 6.9. This is slightly counter-intuitive, but can be understood when we look at the lengths of the sentences produced by these two beam sizes. For example, model L6 produces sentences with the average length of 5.33 words with beam size $b = 1$, while with beam size $b = 5$ the average length drops to just 3.79 words. This is because with larger beam sizes the model always picks the most likely sentence and penalizes heavily any word it is unsure of. This results in the model generating very generic sentences, such as *"SOMEONE looks at SOMEONE"*, over more descriptive ones. This trend of larger beam sizes performing worse is observed with all the LSMDC models L1–L6, but they are not presented in Table 6.9 in the interest of brevity.

6.3.2.2 Results from the LSMDC 2015

The submissions made to the LSMDC 2015 were evaluated using both the automatic metrics and human judgments. However, only the human evaluation was used as the criteria to finally rank the teams. Human evaluations were collected by showing some human judges a video together with five associated captions and asking them to rank the five captions, based on four different criteria:

- *Correctness*: Content in the caption is more correct with the video.
- *Grammar*: Ranking the fluency and readability of the caption.
- *Relevance*: Which caption contains references to more salient items in the video.
- *Helpfulness for the blind*: How helpful is the caption for a blind person to understand the scene.

The five captions consisted of one caption from each of the four submissions and the reference caption for that video.

Table 6.10 presents the automatic evaluation metrics on the blind test set for the four LSMDC submissions. Our model L6 was ranked third among the four teams as per the automatic metrics. We should, however, note here that the evaluation metrics are particularly unreliable on the LSMDC dataset. This is due to having only a single reference caption for the evaluation and also the relatively poor match between the reference captions and the video content as discussed in Section 6.3.1.1.

This is illustrated when we look at the average ranking of the LSMDC submissions as per the four human judgment criteria presented in Table 6.11. Based on human judgments, our submission won the LSMDC 2015 by obtaining the best average ranking in three of the four criteria, and finished second in the *Helpfulness for the blind* criteria. Surprisingly, we also see that three out of the four models outperform the reference captions on the *Grammar* metric. Otherwise, there is still a big gap between the reference caption and the automatic captioning models in the three semantic metrics. A more detailed discussion on the LSMDC results is presented in [51].

Team	BLEU-4	METEOR	ROUGE-L	CIDEr
Visual labels [49]	**0.009**	**0.071**	**0.164**	**0.112**
S2VT [64]	0.007	0.070	0.161	0.091
L6	0.006	0.061	0.156	0.090
Temporal attention [73]	0.003	0.052	0.134	0.062

Table 6.10: LSMDC submissions ranked using automatic evaluation metrics.

Team	Correctness	Grammar	Relevance	Helpful for blind
Reference Caption	1.88	3.13	1.56	1.57
L6	**3.10**	**2.70**	**3.29**	3.29
Temporal attention [73]	3.14	2.71	3.31	3.36
Visual labels [49]	3.32	3.37	3.32	**3.26**
S2VT [64]	3.55	3.09	3.53	3.42

Table 6.11: Human judgment scores for the LSMDC challenge submissions.

6.3.3 MSR-VTT

Similar to the LSMDC dataset, both frame-based and segment-based video features are utilized in the video captioning models presented here for the

#	init	persist	depth	perplex	BLEU-4	METEOR	CIDEr	ROUGE-L
M1	DT	ps-gCNN⊕20Categ	1	27.31	0.396	0.268	0.438	0.588
M2	DT	ps-gCNN⊕20Categ	2-res	27.73	0.409	0.268	0.433	0.598
M3	DT	ps-gCNN⊕20Categ	3-res	28.44	0.370	0.262	0.397	0.575
M4	IDT	ps-gCNN⊕20Categ	2-res	28.13	0.398	0.268	0.432	0.587
M5	DT	c3dfc7	2-res	29.58	0.369	0.268	0.413	0.577
M6	*CNN ensemble of best 4 models*				**0.411**	**0.277**	**0.464**	**0.596**

Table 6.12: Performance of various features and network depths on the validation set of MSR-VTT.

MSR-VTT dataset. However, as the videos in the MSR-VTT dataset are much longer than the videos in the LSMDC dataset, just using features from single keyframe is not sufficient. Therefore, *ps-gCNN* features are extracted on one frame every second in-place of the key-frame based *kf-gCNN*. The *ps-gCNN* features are then pooled using mean pooling.

Apart from the dense trajectory video features, we also experiment with the improved dense trajectory (IDT) and C3D video features on the MSR-VTT dataset. Additionally, we utilize the video category information available for all videos in all splits of the dataset. This information is input to the language model as a one-hot vector of 20 dimensions and is referred to as *20Categ*.

As in the case of LSMDC, we will discuss both the evaluation conducted locally and the results from the video captioning competition conducted based on this dataset, the Microsoft Video to Text Challenge.

6.3.3.1 Results on Validation Set

In order to measure the performance differences due to the different feature combinations, we use the validation set of the MSR-VTT dataset which contains 497 videos. Table 6.12 shows the results on the validation set.

Models M1, M2 and M3 all use the dense trajectory (DT) features as *init* input and the mean-pooled frame-level *ps-gCNN*features concatenated with the video category vector, *20Categ*, as the *persist* input, but they vary in the number of layers in the language model. Comparing their performance we see that the 2-layered model outperforms the single layered model by a small margin, while the 3-layered one is the inferior one.

Model M4 is similar to M2, but uses the improved dense trajectories (IDT) as the *init* input instead. Model M5 differs from M2 by the fact that it uses mean-pooled 3-D convolutional features (C3D) as the *persist* input. We see that both M4 and M5 are competitive, but slightly worse than our best single model, M2. Upon qualitatively analyzing the model outputs, we could see

that each of them performs well on different kinds of videos. For example, model M5, which only uses input features trained for action recognition, does well in videos involving a lot of motion, but suffers in recognizing the overall scenery of the video. Conversely, model M2 trained on frame-level features does better in recognizing objects and scenes, but makes mistakes with the sequence of their appearance, possibly due to the pooling operation. This phenomenon can also be observed in the second row of images in Figure 6.4, where model M5 produces a better caption on the video in the first column, while M2 does better on the video in the second column.

To get maximum utility out of these diverse models, we use the CNN evaluator network to pick the best candidate from the pool of captions generated by the top four models in Table 6.12, M1, M2, M4 and M5. The evaluator is trained using the *ps-gCNN⊕20Categ* as the video feature. This result is shown as model M6 in Table 6.12. We can see that model M6 using the CNN evaluator significantly outperforms, in all the four metrics, every single model it picks its candidates from.

6.3.3.2 Results from the MSR-VTT Challenge

Since the CNN evaluator model, M6, has the best performance on the validation set, it was submitted to the MSR-VTT Challenge. Our submission appears on the leaderboard[4] as *Aalto*. The submissions were evaluated on the test set using the four automatic metrics. The results are shown in Table 6.13. Our submission achieved the best scores in the CIDEr metric and was ranked overall second considering the average ranking across the metrics.

The submissions were also subject to human evaluation as the automatic metrics are known to deviate from human judgments as discussed in Section 3.2.5. The human evaluation was based on three criteria: Coherence (C1), Relevance (C2) and Helpfulness for the blind (C3). But unlike in case of LSMDC, these are collected as scores between zero and five with the latter being the better. Table 6.14 presents the results of the human evaluation. The overall ranking was obtained again by considering the mean ranking across the three metrics. As per human judgments, our submission was ranked the first among the 22 entries in the challenge and thus won the challenge in this category.

Analyzing the two leaderboards, the automatic metric based one and the human evaluation based one, we see that the disagreement between the two is relatively minor, with most teams in the top 10 changing their ranking by only one position. This can most likely be attributed to having a large

[4] `http://ms-multimedia-challenge.com/leaderboard`

number of 20 reference captions per video for the evaluation.

Team	BLEU-4	METEOR	CIDEr	ROUGE-L
v2t_navigator	**0.408**	**0.282**	0.448	**0.609**
Aalto (M6)	0.398	0.269	0.457	0.598
VideoLAB	0.391	0.277	0.441	0.606
ruc-uva	0.387	0.269	**0.459**	0.587
Fudan-ILC	0.387	0.268	0.419	0.595
16 other teams with lower scores appear in the leaderboard				

Table 6.13: Top 5 teams as per automatic evaluation metrics on the MSR-VTT test set.

Team	C1	C2	C3
Aalto (M6)	**3.263**	3.104	**3.244**
v2t_navigator	3.261	3.091	3.154
VideoLAB	3.237	**3.109**	3.143
Fudan-ILC	3.185	2.999	2.979
ruc-uva	3.225	2.997	2.933
16 other teams with lower scores appear in the leaderboard			

Table 6.14: Top 5 teams as per human evaluation.

6.3.4 Qualitative Analysis of Video Captioning

The kind of video captions generated by the models trained on the LSMDC dataset differ from the captions generated on the MSR-VTT data due to the difference in the language used in the reference captions. Figure 6.4 shows some examples (screenshot from the videos) from these two datasets and captions generated by the best models trained on them. The top row contains videos from the LSMDC dataset, while the bottom row shows samples from the MSR-VTT dataset.

The first two images on the top row show examples of accurate captions generated by the L6 model. However, in the video on the right, the model gets confused by the last shot showing a man, and completely ignores the car driving away. This kind of behavior can be seen in many error cases, where the model ignores the main subject of the video and generates the caption based on an object it knows well, e.g. “man” in this case, even if it is insignificant in the video.

In the examples from the MSR-VTT dataset shown in the second row of Figure 6.4, we see three distinct cases. The example on the left shows the case

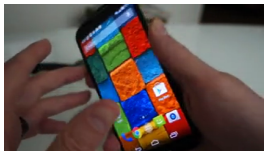
where model M5, based on the C3D features, produces better caption than model M2, which uses frame-level features. The example in the center shows the case where this is reversed and the frame-level features do better than the action-recognition-based C3D features. Finally, the rightmost example shows a case where the CNN evaluator, M6, picks a better caption than the ones generated by both M2, our best single model, and M5.

L6: someone runs up to the car

L6: someone is dancing with her

L6: someone looks up at the house

M6: a man is running in a gym
M2: a man is running
M5: a man is playing basketball



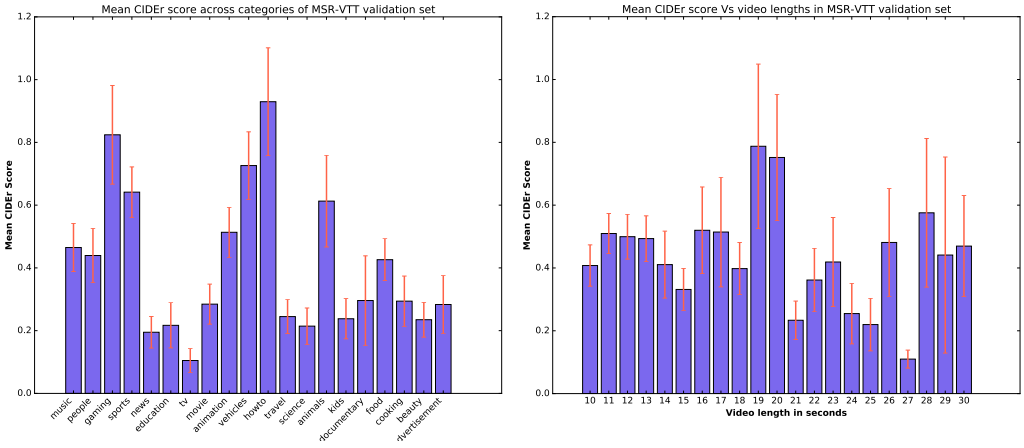
M6: a person is playing with a rubix cube
M2: a man is holding a phone
M5: a person is playing with a rubix cube

M6: cartoon characters are interacting
M2: a person is playing a video game
M5: a group of people are talking

Figure 6.4: Sample captions generated for some test set videos by our models. First row contains captions from model L6 on the LSMDC public test set and the second row contains captions from a few of our best models on samples from MSR-VTT test set.

Next we will analyze how the video captioning performance is affected by video lengths or by video categories. For this we use the MSR-VTT dataset as it has longer videos and also provides the category tag for each video. Figure 6.5 shows the results from this analysis. Here we first evaluate the captions generated by model M6 for each of the 497 videos in the validation set of the MSR-VTT dataset. Then the CIDEr scores for videos belonging to the same category are aggregated and plotted to create Figure 6.5a. Similarly, CIDEr scores for the videos with the same length in seconds (after rounding to nearest integer) is aggregated to produce Figure 6.5b.

From Figure 6.5a we see that the captioning performance varies greatly across categories. The model seems to perform very well on categories such as *how to*, *gaming*, *sports* and *vehicles*. In the case of the *how to* category, this can be attributed to the relatively simplistic nature of the videos. The *how to* videos generally consist of a close-up of a person performing certain distinctive actions, and are thus probably simpler to caption using the action recognition features. A similar structure exists in *gaming* videos, which mostly contain screen captures of people playing video games. We see that



(a) Performance by category *(b) Performance by length*

Figure 6.5: Performance of model M6 as per CIDEr metric compared for various video lengths and sub-categories of the MSR-VTT validation set.

most of the categories where captioning does well have a single visual theme and distinctive category of actions associated with them. In contrast, the categories where the captioning does poorly, e.g. *tv*, *news*, *education*, etc., are unified conceptually and not necessarily visually. For example, the *news* category can contain a variety of scenes, both indoor and outdoor, and a variety of actions. These videos also tend to have many sharp scene changes, thus causing the action recognition features to perform poorly on them.

Interestingly, the length of the video seems to have only a small correlation with the performance as seen in Figure 6.5b. Very long videos (> 20 seconds) seem to perform only slightly worse than the shorter ones. We should also note here that the estimates of performance of the longer videos are noisier, as indicated by the error bars in figure 6.5b, as there are fewer videos falling in these categories. The degradation in longer videos can be expected as the video features we use, both action recognition and frame-based features, do not account for long-term temporal dependencies. Nevertheless, it is still surprising that the degradation is relatively minor.

6.4 Summary of Results and Conclusions

In this chapter, results from the image captioning experiments on the MS-COCO dataset and video captioning experiments on the LSMDC and MSR-VTT datasets were presented. Now we will summarize these results and

highlight the important findings from these experiments.

From the experiments on the MS-COCO dataset, we saw that the proposal to enhance the CNN based *gCNN* image features with explicit object and scene detector features greatly improved the performance. Counter-intuitively, we observe that the use of object location features does not necessarily add to the model: only the smallest of these features, *3+3Gauss*, moderately improved the performance by automatic metrics when used in conjunction with detector features.

Two proposed enhancements to the language model, namely using two separate feature input channels and increasing the depth with residual connections, contribute positively. Using two feature input channels helped utilize more than one image feature and greatly improved the language model performance. Increasing the language model depth moderately improved the performance as per metrics, while it considerably improved the vocabulary and diversity of the captions generated. We also found that the idea of using residual connections between two layers works well even when used with LSTM networks. Using residual connections improved the training convergence speeds of our language model and achieved a lower perplexity measure. However, the attempts to implement a hierarchical decoder in the language model, despite greatly improving the generated caption diversity, affected adversely the correctness of the generated captions and failed to match up to the model with a single stage decoder.

Among the proposed ensembling techniques, CNN-based evaluator network achieved the best performance. In image captioning, while it slightly improved in automatic metrics over the best single model in the ensemble, it contributed positively by increasing the language diversity. Furthermore in the case of video captioning on the diverse MSR-VTT dataset, we found that this ensembling technique clearly outperformed all the models participating in the ensemble and was our best model.

From video captioning experiments, we learned that a combination of action- recognition based and frame-based video features works well for the captioning task. The performance of the video captioning systems is still relatively poorer than that of the image captioning models. Nevertheless, the models presented here for video caption generation can be considered as the state-of-the-art for this task. This is supported by the results from our participation in the LSMDC and MSR-VTT video captioning challenges, both of which we won, as judged by human evaluators.

Chapter 7

Looking Forward

We have, thus far, reviewed the methods and architectures used to build state-of-the-art image and video captioning systems and analyzing their performance qualitatively and quantitatively. Now, it is time to take a step back and determine the progress that has been made and discuss the problems that yet needs to be addressed. In this chapter, we will identify the strengths and weaknesses of the visual captioning systems presented in this thesis. Then, we will discuss a few interesting research problems that will help further improve the visual captioning systems and the more general task of machine understanding of visual media.

7.1 Where Do We Stand?

So far in this thesis we have seen in detail, how image and video captioning systems work, and how well they perform in terms of automatic evaluation metrics, heuristic diversity statistics and human evaluation. We have seen that the image captioning system works relatively better than the video captioning system, partly due to smaller training datasets for video and partly due to deficient feature representations for videos. In particular, video captioning systems perform reasonably well on clips containing less shot boundaries and camera motion, for example in categories like *how-to* and *games* videos on the MSR-VTT dataset. On the other hand they suffer in categories containing complex video styles, for example in the case of *news* one could have multi-pane videos, and *movie* videos usually contain lots of scene changes.

In many cases, captions generated for images are descriptive and accurate, as seen from examples in Appendix A. One can find a good amount of novel captions generated by the models, showing that they not only learn to just

mimic the training data, but also to use the phrases seen in the training set in novel compositions. The generative language model is also able to use correct grammar while describing the visual content, sometimes even better than human captions, as seen in Table 6.11.

Nevertheless, there are still many areas where the captioning models fall short. First and foremost, the vocabulary the models learn to use is very limited. Even our most diverse image captioning model, CNN-ensemble-based C21, is using only 1/8th of the words seen in the training data. This limits the utility of such a model outside of the specific dataset it was trained on. A method to integrate new words into the vocabulary of the model, without re-training it entirely, would be a good extension to make such captioning models viable to use on images and videos in the wild.

Another major bottleneck hindering the progress of captioning systems is the lack of effective and efficient methods to evaluate them. Both kinds of automatic evaluation metrics, ones adopted from machine translation and ones devised specifically for this task, fall short in matching up to human judgment of the quality of the captions generated. These metrics perform better when they have access to a larger number of reference captions, but those are expensive to collect. Alternatively, if one relies solely on human judgments, it still is a tedious and expensive process to get captions from every variant of an algorithm evaluated.

From an algorithm design perspective, despite their quite impressive performance, a drawback of the LSTM-based captioning system is that the interpretability of these models is low. Specifically, it is not apparent how much the visual features are responsible for the generation of a specific word and how much it is caused solely by the bias in the language model. This interpretability becomes especially important when aiming to diagnose the erroneous captions. For example, when we see a model incorrectly caption an image containing a white fire hydrant as a "red fire hydrant", it is hard to tell if the image feature incorrectly encoded the color as red or if the language model, due to the bias in the training data towards red fire hydrants, overrides the image features to produce the wrong caption.

7.2 Future Directions

Based on the shortcomings just discussed, we now enumerate a few problems worthwhile to explore, in order to address these shortcomings and to further the image and video understanding research. Here we try to define and motivate the problems, but with only a limited discussion on how they could be solved.

7.2.1 Generating Multiple Captions

Any image and video is a rich source of information, and can be described by a multitude of captions emphasizing different aspects in them. Our captioning system only generates a single sentence and could thus be an incomplete description of the visual content. Even when we use beam search to sample multiple captions, all the top-ranked captions in the beam tend to be related to each other, sometimes differing only in grammatical arrangements, and thus still only cover a small portion of the available visual information.

A system which can generate multiple captions, ideally with minimum overlap and maximum coverage of the visual information, would be a welcome extension. One way to design such a system would be to further condition the caption generation with an "information vector" which encodes the visual information already described in previously generated captions. The captioning could also be conditioned on specific bounding boxes instead of the entire image, as done in [27], and produce captions describing different parts of the image. The system [27] was trained using the regional captions present in the Visual Genome dataset [33]. Still, although captioning smaller bounding boxes is a way to generate multiple descriptions, this often leads to mundane captions describing single objects and ignoring the larger context in the image.

7.2.2 Better Video Features

We have seen that the video captioning systems proposed in this thesis perform inferior to the image captioning system. This is especially exacerbated in longer videos and videos with large scene changes, where we see that the captions describe only certain small portions of the video. One important cause is that the video features we have used are not good at capturing long-term dependencies and are not designed to handle abrupt scene changes. Both action-based features, trajectory-based and the C3D features, only recognize short-term actions, due to restrictions on the trajectory length and video-segment length, respectively. The pooling techniques used to combine frame-level features into a single vector also lose all temporal information and thus fail to capture such long- term activities.

Thus, a good video feature encoding mechanism, which preserves long term temporal information and can gracefully handle shot boundaries, is a vital component which could improve the video captioning drastically. This could also enable us to build a captioning system for long videos and even generate detailed summaries of long videos. An attempt to generate descriptive paragraph captions for videos was recently made in [77]. The authors use

two recurrent networks in a hierarchical manner, the lower one to generate the current sentence and the upper one to keep track of the paragraph state. The paragraph network re-initializes the sentence-generating RNN after every sentence, based on the current state of the paragraph. Nevertheless, such an approach is still faraway from a complete video summarizer, which could watch long videos and provide a concise summary of the events in them.

7.2.3 Scene Graph Prediction and Matching

Although captioning is a good way to summarize the visual content in an image or a video for humans, it is not the optimal input representation of such information for machines to process further. This is because, any application which uses this summary of visual information based on captions would also need to be able to correctly parse the caption and extract the needed information. This is not straightforward due to the inherent ambiguities in the natural language.

A computationally more useful and less ambiguous representation of the visual information is in the form of scene graphs. Scene graphs consists of nodes which represent the entities present in the scene and edges which encode the relationship between these entities. The nodes can also have a list of attributes associated with them, encoding information such as the color, position, etc. Thus, instead of describing an image using a natural language caption, we could represent it using a scene graph. Unlike natural language descriptions, a scene graph representation would be unique for a set of objects and attributes. This allows further processing stages to easily parse the requisite information. Such a representation could also be used to easily aggregate explicit knowledge about objects and their most characteristic attributes.

Recently, in [53] a method to generate scene graphs from textual description of images was presented. There have also been attempts retrieve images using scene graphs as input in [28]. The authors of [28] also show that scene-graph-based retrieval is more accurate than using captioning methods to rank and retrieve images directly from this textual description. In general, the problem of generating scene graphs taking images as input is still unsolved.

Chapter 8

Conclusions

We have discussed the prospects and problems of automatic image and video captioning in this thesis. We motivated the task of caption generation as a step-up from the simpler image classification problem and as a move towards more complete understanding of visual media. Captioning is also a good task to measure the progress in both visual feature extraction and language generation research.

After identifying the basic building blocks of a captioning system, we reviewed several relevant methods from the literature. The popular encoder–decoder based captioning pipeline presented in [65] was chosen as the baseline model and several extensions were proposed to improve its performance in this thesis. We also highlighted the complexities involved in evaluating a caption generation method and discussed four popular evaluation metrics used in the literature.

Based on the results showing transferability of features from deep convolutional neural networks (CNN) trained for image classification, the vector of activations from the GoogLeNet network was adopted as the primary image feature vector to be used in the image captioning models presented in this thesis. It was also shown that augmenting these CNN features with explicit object and scene detector features greatly improves the captioning performance. In the case of videos, owing to the lack of a dominant feature representation, a few different video feature extraction techniques were experimented with. Three video features aimed at action recognition were utilized in different experiments, namely the dense trajectories, improved dense trajectories and C3D features. Additionally, frame-level features extracted using image CNNs and object detectors were also utilized and it was shown that the best results are obtained when combining the two feature extraction paradigms.

The baseline language model was extended by adding an additional input

channel and by increasing the model's depth. Both these extensions were shown to improve the overall performance. The experiments presented in this thesis also demonstrated that the recently introduced residual connections are effective even when adapted to the Long-Short Term Memory (LSTM) networks. The residual connections help achieve lower training and validation perplexities, and also improve the training speeds.

An efficient ensembling technique based on an evaluator network was also presented. The evaluator network utilizes a CNN sentence encoder and a similarity measure to pick the best caption from a pool of candidates generated by multiple language models. This ensembling technique was shown to work best in video captioning of the MSR-VTT dataset, wherein the language models participating in the ensemble were good at different parts of the dataset. In the image captioning task on the COCO dataset, using the CNN based evaluator only moderately improves the performance as per the automatic metrics, but it greatly increases the diversity of the generated captions.

Utilizing the video captioning methods presented here, we participated in two video captioning competitions namely the LSMDC and MSR-VTT challenges. Our methods won both the competitions as judged by human evaluators. The work presented in this thesis has also led to three publications [55–57].

To put the current state of the captioning systems into context, we can draw a layman's analogy to the progression of visual description capabilities in humans. When we learn to speak, in the first stage we learn to utter only a few independent words. In the second stage, we learn to repeat and recompose a few sentences heard from our teachers. Only in the later stages we learn to precisely reason about every word we utter and can invent novel descriptions or write long essays about a single image. We see the current image and video captioning systems to be in the second stage, wherein they tend to repeat and in some cases recompose parts of the captions they have seen in the training set. There is still much work to be done before machines can generate well-reasoned novel descriptions and maybe even thousand word essays describing the content of visual input.

Bibliography

[1] *Hours of video uploaded to YouTube every minute as of July 2015*, Accessed June 23, 2016. `http://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/`.

[2] *Instagram Statistics*, Accessed June 23, 2016. `https://www.instagram.com/press`.

[3] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv abs/1409.0473* (2014).

[4] Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[5] Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. A maximum entropy approach to natural language processing. *Computational Linguistics 22* (1996).

[6] Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. Class-based n-gram models of natural language. *Computational Linguistics 18* (1992).

[7] Chang, C.-C., and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011).

[8] Chen, D. L., and Dolan, W. B. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the ACL Conference on Human Language Technologies* (2011), Association for Computational Linguistics.

[9] Chen, X., Hao Fang, T.-Y. L., Vedantam, R., Gupta, S., DollÃ¡r, P., and Zitnick, C. L. Microsoft COCO captions: Data collection and evaluation server. *arXiv abs/1504.00325* (2015).

[10] Cortes, C., and Vapnik, V. Support-vector networks. *Machine Learning 20* (1995).

[11] Cui, Y., Ruggero Ronchi, M., and Lin, T.-Y. 1st captioning challenge slides. Large-scale Scene Understanding Workshop, 2015.

[12] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2009).

[13] Denkowski, M., and Lavie, A. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation* (2014), ACL.

[14] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015).

[15] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the International Conference on Machine Learning* (2014).

[16] Fang, H., Gupta, S., Iandola, F., Srivastava, R., Deng, L., Dollar, P., Gao, J., He, X., Mitchell, M., Platt, J., Zitnick, L., and Zweig, G. From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015).

[17] Farhadi, A., Hejrati, M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J., and Forsyth, D. Every picture tells a story: Generating sentences from images. In *European Conference on Computer Vision* (2010).

[18] Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Mikolov, T., et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems* (2013).

[19] Gong, Y., Wang, L., Guo, R., and Lazebnik, S. Multi-scale orderless pooling of deep convolutional activation features. *arXiv abs/1403.1840* (2014).

[20] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision* (2015).

[21] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition* (2016).

[22] Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA*: Neural Networks for Machine Learning, 2012.

[23] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Computing 9* (1997).

[24] Hodosh, M., Young, P., and Hockenmaier, J. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research 47* (2013).

[25] Ji, S., Xu, W., Yang, M., and Yu, K. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013).

[26] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. Caffe: Convolutional architecture for fast feature embedding. *arXiv abs/1408.5093* (2014).

[27] Johnson, J., Karpathy, A., and Fei-Fei, L. Densecap: Fully convolutional localization networks for dense captioning. *arXiv abs/1511.07571* (2015).

[28] Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D. A., Bernstein, M. S., and Fei-Fei, L. Image retrieval using scene graphs. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2015), IEEE.

[29] Karpathy, A., and Fei-Fei, L. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015).

[30] Karpathy, A., Joulin, A., and Fei-Fei, L. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in neural information processing systems* (2014).

[31] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014).

[32] Kim, Y. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2014), Association for Computational Linguistics.

[33] Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., Bernstein, M., and Fei-Fei, L. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv abs/1602.07332* (2016).

[34] Krizhevsky, A., Sutskever, I., and Hinton, G. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012).

[35] Kulkarni, G., Premraj, V., Ordonez, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., and Berg, T. L. Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35* (2013).

[36] Li, S., Kulkarni, G., Berg, T. L., Berg, A. C., and Choi, Y. Composing simple image descriptions using web-scale n-grams. In *Proceedings of the Conference on Computational Natural Language Learning* (2011).

[37] Li, X., Snoek, C. G. M., Worring, M., Koelma, D. C., and Smeulders, A. W. M. Bootstrapping visual categorization with relevant negatives. *IEEE Transactions on Multimedia 15* (2013).

[38] Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. *Text summarization branches out: Proceedings of the ACL workshop 8* (2004).

[39] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., and Zitnick, C. L. Microsoft COCO:

Common objects in context. In *Proceedings of the European Conference on Computer Vision* (2014).

[40] Maron, O., and Lozano-Pérez, T. A framework for multiple-instance learning. *Advances in neural information processing systems* (1998).

[41] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. Recurrent neural network based language model. In *Interspeech* (2010).

[42] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013).

[43] Norouzi, M., Mikolov, T., Bengio, S., Singer, Y., Shlens, J., Frome, A., Corrado, G. S., and Dean, J. Zero-shot learning by convex combination of semantic embeddings. *arXiv abs/1312.5650* (2013).

[44] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting on Association for Computational Linguistics* (2002), ACL.

[45] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2014).

[46] Perronnin, F., Sánchez, J., and Mensink, T. Improving the fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision* (2010).

[47] Rashtchian, C., Young, P., Hodosh, M., and Hockenmaier, J. Collecting image annotations using amazon's mechanical turk. In *Procedings of the NAACL Conference on Human Language Technologies* (2010).

[48] Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv abs/1506.01497* (2015).

[49] Rohrbach, A., Rohrbach, M., and Schiele, B. The long-short story of movie description. In *Proceedings of the German Conference on Pattern Recognition* (2015), Springer.

[50] Rohrbach, A., Rohrbach, M., Tandon, N., and Schiele, B. A dataset for movie description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015).

[51] Rohrbach, A., Torabi, A., Rohrbach, M., Tandon, N., Pal, C. J., Larochelle, H., Courville, A. C., and Schiele, B. Movie description. *arXiv abs/1605.03705* (2016).

[52] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* (2015).

[53] Schuster, S., Krishna, R., Chang, A., Fei-Fei, L., and Manning, C. D. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Workshop on Vision and Language* (2015), ACL.

[54] Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2014).

[55] Shetty, R., and Laaksonen, J. Video captioning with recurrent networks based on frame-and video-level features and visual content classification. *IEEE International Conference on Computer Vision Workshop on LSMDC abs/1512.02949* (2015).

[56] Shetty, R., and Laaksonen, J. Frame- and segment-level features and candidate pool evaluation for video caption generation. In *Proceedings of the ACMMM Multimedia Grand Challenge Solutions* (2016).

[57] Shetty, R., R-Tavakoli, H., and Laaksonen, J. Exploiting scene context for image captioning. In *ACMMM Vision and Language Integration Meets Multimedia Fusion Workshop* (2016).

[58] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv abs/1409.1556* (2014).

[59] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015).

[60] Torabi, A., Chris, P., Hugo, L., and Aaron, C. Using descriptive video services to create a large data source for video annotation research. *arXiv abs/1503.01070* (2015).

[61] Tran, D., Bourdev, L. D., Fergus, R., Torresani, L., and Paluri, M. C3D: Generic features for video analysis. *arXiv abs/1412.0767* (2014).

[62] Vedaldi, A., and Zisserman, A. Efficient additive kernels via explicit feature maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2010).

[63] Vedantam, R., Lawrence Zitnick, C., and Parikh, D. Cider: Consensus-based image description evaluation. In *The IEEE Conference on Computer Vision and Pattern Recognition* (2015).

[64] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., and Saenko, K. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision* (2015).

[65] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015).

[66] Wang, H., Kläser, A., Schmid, C., and Liu, C. Action recognition by dense trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2011), IEEE Computer Society.

[67] Wang, H., and Schmid, C. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision* (2013).

[68] Weston, J., Bengio, S., and Usunier, N. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning 81* (2010).

[69] Xiao, J., Ehinger, K. A., Hays, J., Torralba, A., and Oliva, A. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision* (2014).

[70] Xiao, J., Hays, J., Ehinger, K., Oliva, A., and Torralba, A. SUN database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2010).

[71] Xu, J., Mei, T., Yao, T., and Rui, Y. MSR-VTT: A large video description dataset for bridging video and language. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016).

[72] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. *arXiv abs/1502.03044* (2015).

[73] Yao, L., Torabi, A., Cho, K., Ballas, N., Pal, C., Larochelle, H., and Courville, A. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE International Conference on Computer Vision* (2015).

[74] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems* (2014).

[75] You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. Image captioning with semantic attention. *arXiv abs/1603.03925* (2016).

[76] Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics 2* (2014).

[77] Yu, H., Wang, J., Huang, Z., Yang, Y., and Xu, W. Video paragraph captioning using hierarchical recurrent neural networks. *arXiv abs/1510.07712* (2015).

[78] Zaremba, W., Sutskever, I., and Vinyals, O. Recurrent neural network regularization. *arXiv abs/1409.2329* (2014).

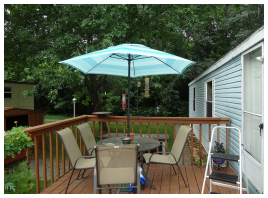
[79] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems* (2014).

Appendix A

Samples from COCO dataset

A few examples from the COCO validation set are shown with the captions generated by models C27 (ensemble) and C19. The examples are randomly chosen among the images where the two models produce different captions.

			
C27:	a man riding a motorcycle down the road	a group of people sitting at tables with laptops	a person holding a hot dog on a bun
C19:	a man riding a motorcycle down a street	a group of people sitting around a table	a close up of a person holding a hot dog
			
C27:	a person on a skateboard does a trick	a kitchen filled with lots of metallic appliances	a man riding a horse down a dirt road
C19:	a person riding a skateboard on a ramp	a kitchen with lots of appliances in it	a herd of zebra walking on across a dirt road
			
C27:	a close up of a giraffe looking at the camera	a yellow school bus parked in a parking lot	a rear view mirror of a car in a car
C19:	a giraffe standing next to a wooden fence	a yellow and yellow bus on a city street	a view from the side of a car window

			
C27:	a group of people standing around a luggage carousel	a patio area with chairs and a umbrella	a group of people standing on top of a sandy beach
C19:	a group of people standing in a room	an outdoor patio with chairs and chairs	a group of people on a beach flying kites
			
C27:	two women standing next to each other holding tennis racquets	a person holding a cell phone in their hand	a group of people holding umbrellas standing in the rain
C19:	a couple of women standing on a tennis court	a close up of a person holding a cell phone	a group of people walking down a street holding umbrellas
			
C27:	a motorcycle that is parked in the grass	a group of men on a field playing baseball	a cat lying on a bed with a stuffed animal
C19:	a motorcycle is parked in a grassy field	a baseball player swinging a bat at a ball	a cat lying on top of a stuffed animal
			
C27:	a row of chairs and chairs on a beach	a group of people playing a game of tennis	a brown and white horse standing in a field
C19:	a table with chairs and chairs on it	a group of young people playing a game of frisbee	a horse that is standing in the grass